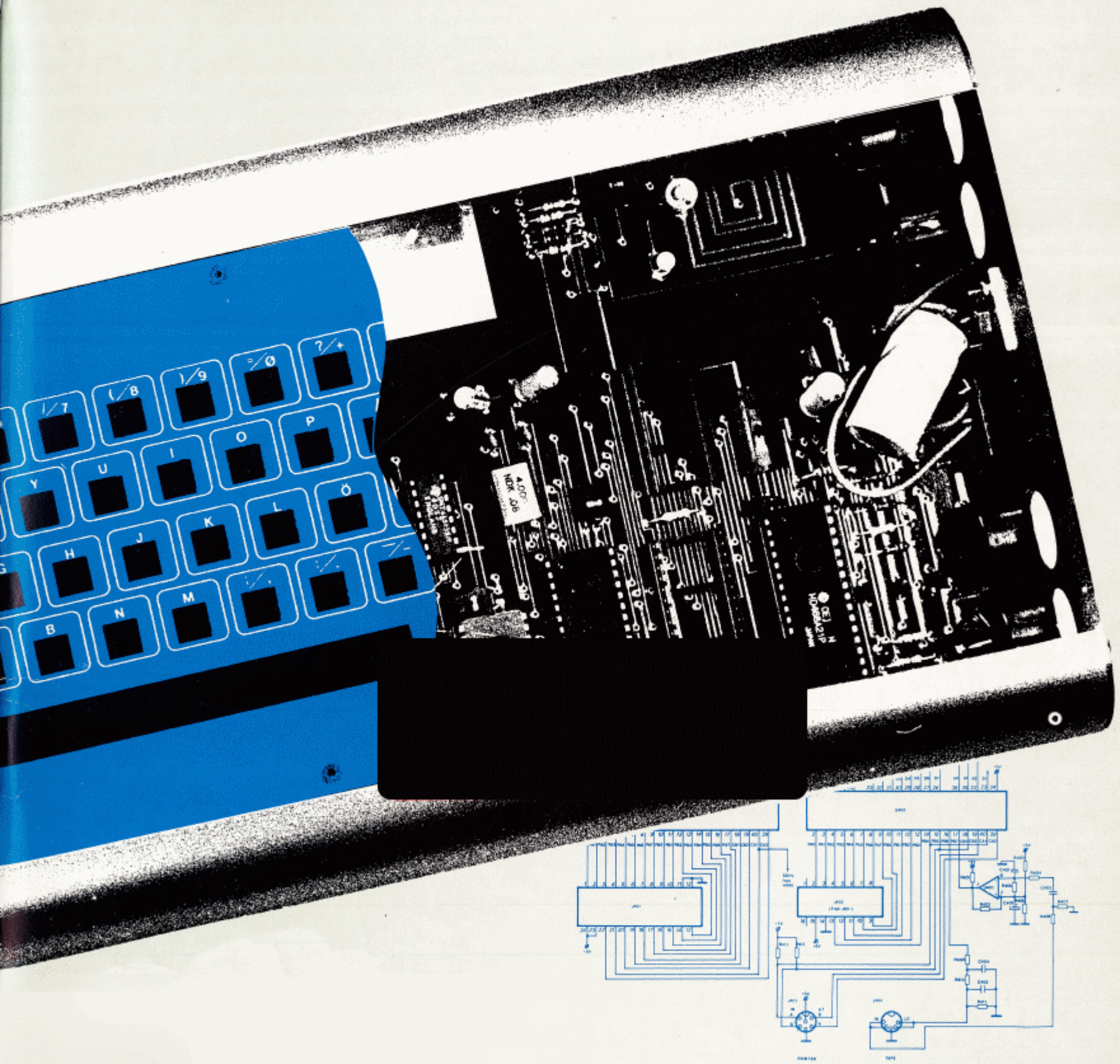
TEMAL 100

Teknisk manual
ESSELTE 100



Anders Andersson
Lars Björklund

TEMAL 100

Teknisk manual
ESSELTE 100

Anders Andersson **Lars Björklund**

ESSELTE STUDIUM

TILL LÄSAREN

Eftersom det största användningsområdet för skoldatorn Esselte 100 torde vara grundläggande undervisning i datalära och programmering, är den manual som bipackas maskinen främst inriktad på att beskriva handhavandet vid normal programmering i Basic.

Men Esselte 100 konstruerades även för att användas vid mer avancerade tekniska tillämpningar, och det är denna sida av maskinen som presenteras här. Vårt mål har inte varit att skriva en komplett kurs i konstruktion av mikrodatorer, utan TEMAL 100 skall snarare ses som en uppslagsbok för användare som redan besitter grundläggande kunskaper i mikrodatorteknik. Dock har vi försökt göra vissa kapitel lättillgängliga genom att belysa med praktiska exempel och ge förslag på kopplingsscheman för enklare tillbehör.

Anders Andersson **Lars Björklund**

INNEHÅLLSFÖRTECKNING

Kapitel 1

HÅRDVARUUPPBYGGNAD
1.1 Specifikation 1
1.2 CPU 1
1.3 Minnen 2
1.4 Strömförsörjning och HF 2
1.5 I/O 3
1.6 Video 3
1.7 Tangentbordet 4
1.8 Stycklista 5
1.9 Komponentplacering 6

Kapitel 2

BASIC-TOLKEN
2.1 Allmänt 9
2.2 Minnesplan 9
2.3 Var och hur lagras Basic-program? 11
2.4 Vad händer vid start? 11
2.5 Avbrott 12
2.6 Exekveringshastighet 12
2.7 Att hushålla med minnet 12
2.8 Systemswitchar 13

Kapitel 3

ESSELTE 100 SOM SYSTEMKOMPONENT
3.1 Inledning 15
3.2 Portsystemet 15
3.3 Möjligheter tack vare portsystemet 15
3.3.1 PROM-kassett 16
3.3.2 Yttre terminal 16
3.3.3 MASTER 16

Kapitel 4

BILDSKÄRMEN OCH TANGENTBORDET
4.1 Bildskärmen 17
4.2 Markören 17
4.3 Extra funktioner 17
4.4 Tangentbordet 18
4.5 Att hämta information från tangentbordet 18
4.6 Gemener 19
4.7 Att skydda fält på bildskärmen 19
4.8 Texthantering 19
4.9 Att blanda gemener och grafik 20
4.10 Några speciella funktioner 20

Kapitel 5

PRINTER OCH TERMINAL
5.1 Format 21
5.2 Hastighet 21
5.3 Optioner 21
5.4 Exempel på printrar 21
5.4.1 Trendcom/Esselte 21
5.4.2 Texas Silent 21
5.5 Interface-kontakten 21
5.6 Esselte 100 som terminal 22
5.6.1 TERM 22
5.6.2 Formatändring 22
5.6.3 Några tips 22
5.6.4 Några exempel 23

Kapitel 6

SAMMANKOPPLING AV FLERA ESSELTE 100
6.1 Esselte 100 som terminal till en annan Esselte 100 25
6.2 Lärarslinga 25
6.2.1 Hårdvara 25
6.2.2 Användning 25
6.2.3 Några kommentarer 26
6.2.4 Ökning av överföringshastigheten 26
6.2.5 Överföring av program 26

Kapitel 7

BANDSPELAREN
7.1 Format 27
7.2 Vad sänds ut? 27
7.3 Att lagra annat än program 27
7.4 Att styra bandspelaren 27

Kapitel 8

GRAFIK
8.1 Grafik på bildskärmen 29
8.1.1 Speciella grafikkommandon 29
8.2 Grafiska tecken 30
8.3 Att rita kurvor 31
8.4 Att skriva med stora tecken 32
8.5 Att skriva med stora egna tecken 32
8.6 Rörlig grafik 34

Kapitel 9

MASKINSPRÅKSPROGRAMMERING
9.1 Varför maskinkod? 37
9.1.1 Minneskarta 37
9.1.2 Maskinvara 37
9.2 Hur man anropar maskinkodsprogram 37
9.2.1 Anrop från Basic-program 37
9.2.2 A=USER(X) 37
9.2.3 X=SYS(Y,Z) 37
9.2.4 A=INP(X) 38
9.2.5 OUT(X,Y)38
9.2.6 IO 38
9.2.7 APULS och BPULS 38
9.2.8 Egna OUT-rutiner 38
9.2.9 TIME(X) 38
9.3 Maskinkodsmonitorn 38
9.3.1 Sammanfattning av monitorkommandona 39
9.3.2 Var får du skriva maskinkod? 39
9.3.3 Jämförelse mellan Basic och maskinkod 39
9.4 Att spara maskinkodsprogram på kassett 39
9.5 Subrutiner i Basic-tolken 40
9.5.1 Utmatningsrutiner 40
9.5.2 Inhämtning av data 40
9.5.3 Grafiska rutiner 40
9.5.4 Extra funktioner 40
9.5.5 Beräkningsrutiner 41

Kapitel 10

DATABASER
10.1 Databashantering med Esselte 100 43
10.2 Att lägga upp register 43
10.3 Mätvärdesinsamling 44
10.4 BAS 100, ett databashanterings-program 44

Kapitel 11

DIGITALA IN- OCH UTSIGNALER
11.1 Digitala signaler via PRINTER-kontakten 47
11.2 Extra parallellport 48
11.3 Realtidsklockan 48
11.4 Realtidsklockan som stoppur 49

Kapitel 12

MÄTNING OCH STYRNING VIA PRINTERKONTAKTEN
12.1 Serie- och parallellgränssnitt 51
12.2 PRINTER-kontakten 51
12.3 Tidmätning 51
12.3.1 Tidmätningsinstruktioner 51
12.3.2 Gränssnitt för tidmätning 52
12.3.3 Programexempel 52
12.4 Frekvensmätning 53
12.4.1 Frekvensmätningsinstruktioner 53
12.4.2 Gränssnitt för frekvensmätning 53
12.4.3 Programexempel 53
12.5 Mätvärdesomvandlare 53
12.5.1 Resistans/frekvens-omvandlare 53
12.5.2 Spänning/frekvens-omvandling 54
12.5.3 Pulsbreddsomvandling 55
12.5.4 Joy-stick 56
12.5.5 Exempel på mätsystem 56
12.6 Analoga signaler 57
12.6.1 Pulsbreddsmodulerad D/A-omvandlare 57
12.6.2 Resistansnät 58
12.7 IO 100, ett digitalt gränssnitt till Esselte 100 58

APPENDIX
MC 6800 Instruction set 59

REGISTER 62

Kapitel 1

Hårdvaruuppbyggnad

1.1 Specifikation

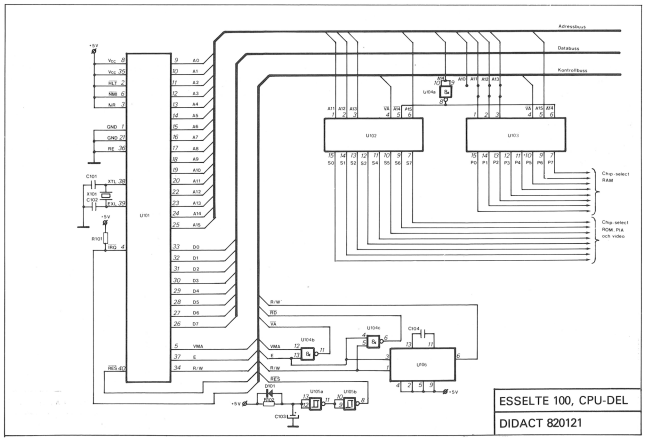
Esselte 100 är uppbyggd kring Motorolas 8-bitars mikroprocessor MC6802. Klockfrekvensen är 1 MHz och klockan är kristallstyrd. I systemet ingår totalt 8 Kbyte RAM av s k "Byte-Wide"-typ, varav 1 Kbyte används för bildskärmen ("Video-RAM"). Totalt använder systemet 14 Kbyte ROM (EPROM typ 2716), varav 2 Kbyte används för teckengenerering och 12 Kbyte för operativsystem och Basic-tolk. Vidare ingår som standard en I/O-krets typ MC6821 (PIA) som används till tangentbordet och för anslutning till bandspelare och printer. På kortet finns plats för ytterligare en PIA, men kretsen ingår inte i standardversionen av Esselte 100. Om även denna PIA kopplas in, har användaren tillgång till ytterligare 20 datakanaler in och ut, att användas efter eget gottfinnande! Vidare kan kretsen utnyttjas som realtidsklocka. Inkopplingen av den extra PIA:n beskrivs i kapitel 11.

Även utan den extra PIA:n erbjuder Esselte 100 mycket stora möjlighter som mätdator och för styr- och reglerändamål, tack vare att Basic-tolken kan utnyttja printerporten som in- och utport. Mera om detta i kapitel 11.

Esselte 100 är uppbyggd på två separata kretskort, förbundna med en flatkabel. Det ena kortet utgörs av tangentbordet, det andra är själva datorn. Båda korten sitter inskjutna i spår i den strängsprutade aluminiumprofil vilken används som apparatlåda.

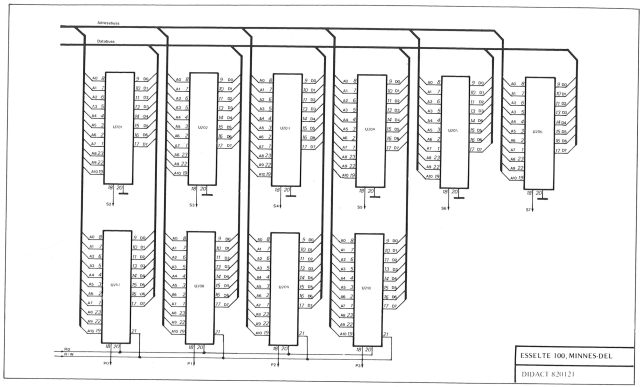
1.2 CPU

Klockan styrs av en 4 MHz kristall (X). Vid start genereras en RESET-signal till CPU och PIA-kretsar med hjälp av ett RC-nät (R102, C103) och en Schmitt-trigger (U105). RAM-kretsarna fordrar en R/W-signal, som slutar >200 ns innan klockpulsen slutar, varför R/W-signalen från 6802 måste modifieras. Detta görs med en monovippa (U106). Som adressavkodare används två stycken SN74LS138 (U102 och U103). U103 används utelslutande för avkodning av RAM-minnena. För att möjliggöra ev framtida RAM-minnesexpansion finns 3 byglingar inlagda på kortet. Möjlighet finns att placera in upp till 48 K RAM. Avkodaren U102 används för PROM, video- och PIA-kretsar. Minnesplan återfinns i avsnitt 2.2, på sidan 9.



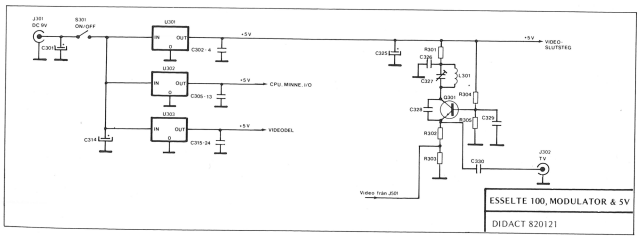
1.3 Minnen

Basic-tolken och operativsystemet är lagrade i 6 st EPROM, U201—U206, typ 2716 (totalt 12 K). Som RAM används MK4118 alt HMS6116 (olika kombinationer kan förekomma), U207—U210, totalt 8 K RAM.



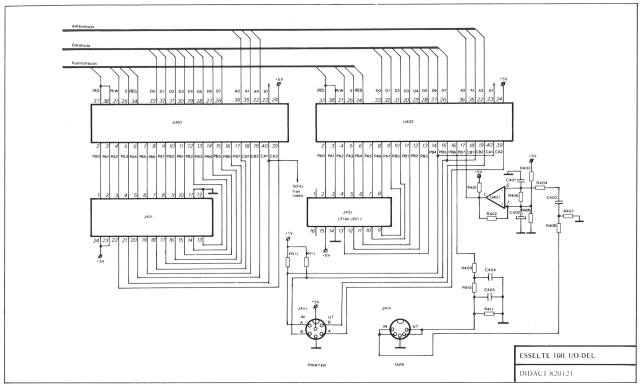
1.4 Strömförsörjning och HF

Esselte 100 strömförsörjs från en yttre batterieliminator Mascot 696, som lämnar en stabiliserad likspänning på 9 V vid 0,8 A. Kretsarna och datorn fordrar en stabiliserad likspänning, 5 V ±5%. För att minimera inverkan av störningar på matningsspänningen utnyttjas tre separata regulatorer (U301, U302 och U303). U 302 strömförsörjer CPU, minnen och PIA-kretsar, U303 videodelen (utom videoslutsteget) och U301 videoslutsteg och VHF-modulator. Den stabiliserade 5 V-spänningen finns tillgänglig på printerkontakten och kan användas för strömförsörjning av t ex mätvärdesomvandlare eller andra yttre enheter. Nätaggregatet till Esselte 100 är dimensionerat för ett maximalt extra strömuttag på 50 mA till yttre enheter. VHF-modulatorn utgörs av en amplitudmodulerad oscillator, som sänder på ca 60 MHz (kanal 3). I vissa fall (t ex om den lokala TV-sändaren utnyttjar kanal 3), kan det vara lämpligt att ändra oscillatorfrekvensen. Detta kan göras med C327.



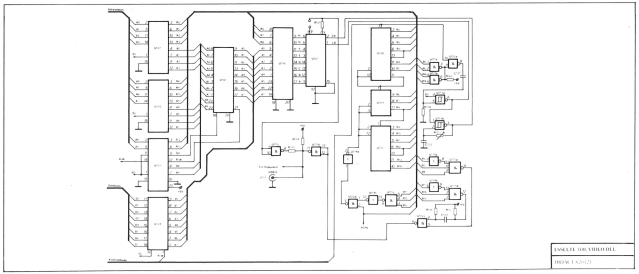
1.5 I/O

För kommunikation med tangentbord, bandspelare, printer etc används en PIA, MC6821 (U402). Kassettinterfacet utgörs av en analog krets, U403. Tangentbordet ansluts via en 16-polig flatkabel till en kontakt på huvudkortet, J402. Plats finns för ytterligare en PIA (U401) där yttre enheter kan anslutas till en 24-polig kontakt (J401) på huvudkortet.



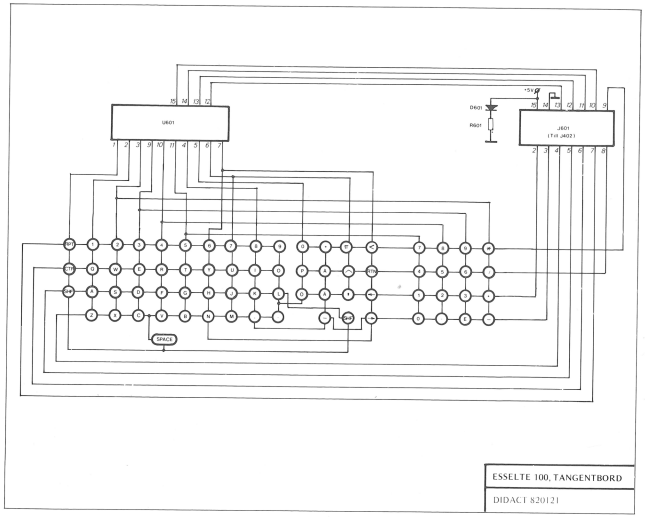
1.6 Video

Ytterligare 1 K RAM (U505) används som bildskärmsminne. De 1 024 positionerna är organiserade som 32 rader om vardera 32 tecken. Teckengeneratorn består av räknare (U508—U510), ett skiftregister (U507) och ett tecken-PROM typ 2716 (U506) där mönster för alla bokstäver och tecken samt grafiska symboler finns lagrade (se Teckenkarta, sid 30). Skiftregistret klockas av en oscillator (U513b) vars frekvens kan justeras med trimpotentiometern R503. Justeringen påverkar teckenbredden. Vidare ingår logik för generering av bild- och linjesynkpulser. Dessa signaler blandas i U511 och resultatet blir en sammansatt videosignal som dels styr VHF-modulatorn, dels matas ut på videokontakten.



1.7 Tangentbordet

Tangentbordet är uppbyggt på ett separat kort och ansluts till huvudkortet med en flatkabel. Det består av 72 st en-poliga slutande tryckknappar, arrangerade i ett matrismönster. Via PIA-kretsen U402 matas binär information ut till en avkodare (U601), som sekventiellt pekar ut en av 10 rader åt gången, 100 ggr/sekund. Om en tangent är nedtryckt tar datorn reda på vilken rad och vilken kolumn tangenten tillhör.



1.8 Stycklista

ESSELTE 100, komponentförteckning

Kretsar:

U101	MC6802	U208	HM6116P-3	U503	SN74LS157
U102	SN74LS138		alt MK4118	U504	SN74LS245
U103	SN74LS138	U209	HM6116P-3	U505	MK4118
U104	SN74LS00		alt MK4118	U506	2716
U105/513	SN74LS132	U210	HM6116P-3	U507	SN74LS165
U106	SN74LS122		alt MK4118	U508	SN74LS393
		U301	MC78L05CP	U509	SN74LS93
U201	2716	U302	MC7805CT	U510	SN74LS393
U202	2716	U303	MC7805CT	U511	SN7403
U203	2716			U512	SN74LS00
U204	2716	U401	MC6821 (option)	U513/105	SN74LS132
U205	2716	U402	MC6821	U514	SN74LS04
U206	2716	U403	LM339N	U515	SN74LS10
U207	HM6116P-3				
	alt MK4118	U501	SN74LS157	U601	SN74LS145
		U502	SN74LS157		

Transistor:

Q301	2N2369

Kondensatorer:

C101	27 pF keramik	C311	100 nF keramik	C325	4,7 µF/16 V ellyt
C102	27 pF keramik	C312	100 nF keramik	C326	100 nF keramik
C103	4,7 µF/16 V ellyt	C313	100 nF keramik	C327	3,5–13 pF trim
C104	39 pF keramik	C314	4,7 µF/16 V ellyt	C328	6,8 pF keramik
C301	2 200 µF/16 V ellyt	C315	100 nF keramik	C329	22 nF keramik
C302	100 nF keramik	C316	100 nF keramik	C330	2,2 pF (strökap)
C303	100 nF keramik	C317	100 nF keramik	C401	470 pF keramik
C304	100 nF keramik	C318	100 nF keramik	C402	47 µF/16 V ellyt
C305	100 nF keramik	C319	100 nF keramik	C403	22 nF keramik
C306	100 nF keramik	C320	100 nF keramik	C404	10 nF keramik
C307	100 nF keramik	C321	100 nF keramik	C405	10 nF keramik
C308	100 nF keramik	C322	100 nF keramik	C501	220 pF keramik
C309	100 nF keramik	C323	100 nF keramik	C502	68 pF keramik
C310	100 nF keramik	C324	100 nF keramik	C503	68 nF plastfolie

Dioder:

D101	1N4148
D601	LD30 lysdiod

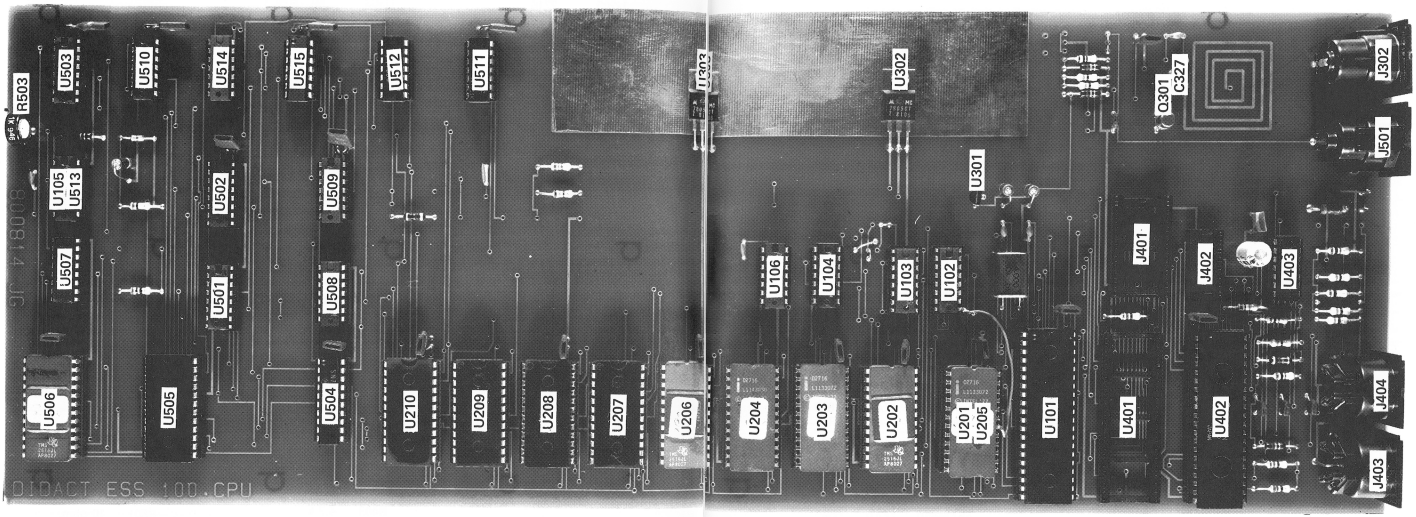
Motstånd:

R101	3,3 k	R402	1 M	R413	3,3 k
R102	100 k	R403	4,7 k	R501	10 ohm
R301	270 ohm	R404	4,7 k	R502	1 k
R302	2,2 k	R405	100 k	R503	1 k trim
R303	270 ohm	R406	10 k	R504	150 ohm
R304	4,7 k	R407	270 ohm	R505	27 ohm
R305	4,7 k	R408	10 k	R506	10 k
R401	4,7 k	R409	10 k	R507	1 k
		R410	47 k	R508	270 ohm
		R411	4,7 k	R601	150 ohm
		R412	3,3 k		

Kristall:

X101	4 MHz

1.9 Komponentplacering



Kapitel 2

Basic-tolken

2.1 Allmänt

Basictolken är ett program som ligger i PROM. När data matas in från tangentbordet, lagras det i en radbuffert.

Vid inmatning lagras alla karaktärer sekventiellt, utom de tre kontrolltecknen CTRL-H, CTRL-X och CTRL-M. Dessa tecken har till funktion att editera inmatningen och deras funktion utförs direkt då de påträffas.

CTRL-H eller Backspace medför att föregående tecken suddas ur radbufferten.

CTRL-X eller Delete medför att hela bufferten suddas.

CTRL-M eller Carriage return (RETURN) är det tecken som avslutar en inmatning till radbufferten. När detta tecken inmatas kommer Basictolken att börja tolka det meddelande som avgivits.

Om det i början av radbufferten finns ett radnummer, kommer hela radbufferten att flyttas ut i primärminnet. Det placeras in på första lediga minnesplats eller på den plats som radnumret angivit. En viss omflyttning i minnet kan alltså ske, så att t ex rad 15 kan placeras in mellan rad 10 och 20. Om radnummer saknas, kommer tolken att försöka utföra den instruktion eller det kommando som finns lagrat. Detta sker genom att tolken genomsöker sina kommandotabeller för att hitta adresser till de funktioner som angivits. Finns ej kommandot kommer tolken att ge en felutskrift.

Ett av de kommandon som finns, 'RUN', medför att tolken börjar läsa i minnet och utföra de instruktioner som finns där i nummerordning.

2.2 Minnesplan

Blockschema

Adress Hex		
0000 0036	RAM	Egna maskinkodsprogram
0037 0064	RAM	Operativsystem
0065 0286	RAM	Basictolken (Register och buffert)
0287 1EFF	RAM	Basicprogram
1F00 1FFF	RAM	Stack
2000 BFFF		Ej avkodat
C000 C3FF	RAM	Bildskärm
C400 C7FF		Ej avkodat
C808 C80B	PIA1	Tangentbord mm
C810 C813	PIA2	Option
C814 CFFF		Ej avkodat
D000 FFFF	ROM	Basictolk och operativsystem

Adress Hex	Adress Decimal	Användning	
0000	00000	RAM	Tillgängligt för egna program
0037	00055	RAM	Pekare till monitor och operativsystem
0065	00101	RAM	Pekare och register till Basictolken
007C	00124	RAM	Nästa Basicrad
0080	00128	RAM	Startpekare Basicprogram
0084	00132	RAM	Högsta radnummer
008E	00142	RAM	Första variabel
0090	00144	RAM	Stränglängd
0096	00150	RAM	Första lediga RAM
00B1	00177	RAM	Här börjar variabeltabellen
00D8	00216	RAM	Pekare till Strängbuffert
00EF	00239	RAM	Nästa i Radbuffert
00F1	00241	RAM	Pekare till Radbuffert
00F3	00243	RAM	Pekare till slut av minne
00F5	00245	RAM	Utadress
00F7	00247	RAM	Inadress
00F9	00249	RAM	Portnummer
00FA	00250	RAM	Antal blanktecken i slut på rad
00FB	00251	RAM	Antal tecken/rad
00FC	00252	RAM	Adress till USER rutin
00FE	00254	RAM	Adress för variabel rutin
0100	00256	RAM	Indexregisterstack
0140	00320	RAM	Variabelstack
01BE	00446	RAM	FOR-stack
023E	00574	RAM	Radbuffert 72 byte
0287	00647	RAM	Basicprogram 7289 byte
1FFF	08191	RAM	Sista RAM positionen = STACK
2000	08192		Ej besatt eller avkodat
C000	49152	RAM	Första pos. i VIDEORAM
C3FF	50175	RAM	Sista pos. i VIDEORAM
C808	51208	PIA	Första pos i PIA1 (Standard)
C810	51216	PIA	Första pos i PIA2 (Option)
D000	53248	ROM	Basic-tolk, kallstart
D003	53251	ROM	—"— varmstart
F833	63515	ROM	PATCH Minimonitor
F121	61729	ROM	Operativsystem
F872	63602	ROM	Resetkallstart
FFFF	65535	ROM	Sista adress

2.3 Var och hur lagras Basic-program

Basicprogrammet lagras i en ordnad följd i RAM-minnet med adress från 0287 Hex och framåt. Vi skall titta på hur ett enkelt program lagras:

Skriv in följande program:

```
1 A=1234
2 PRINT A
```

Gå nu ut i maskinkodsmonitorn med kommandot PATCH (se kap 9). Skriv M 0287, läs vad som står där och gå vidare med mellanslag. När du är nöjd tryck på return och B.

Det du ser bör se ut så här:

Adress	Data	Betyder
0287	00	Radnummer
0288	01	
0289	EC	Kod för Tilldelning (=)
028A	F8	
028B	0C	Antal byte till nästa rad = 12
028C	41	A
028D	3D	=
028E	31	1
028F	32	2
0290	33	3
0291	34	4
0292	00	Slut på raden
0293	00	Radnummer
0294	02	
0295	EC	Kod för PRINT
0296	4A	
0297	07	Antal byte till nästa rad = 7
0298	41	A
0299	00	Slut på raden

Den kod som nämnes är egentligen en adress in i Basic-tolken, där motsvarande instruktion utförs.

Kör nu programmet med RUN, gå in i PATCH och upprepa föregående. Efter programmet har nu variabeln A lagrats enligt:

Adress	Data	Funktion
029A	41	Namn, dvs A
029B	20	Typ av variabel, i detta fall flyttal
029C	01	Talet: 1
029D	23	23
029E	40	4
029F	00	
02A0	00	
02A1	04	10-talsexponent i tvåkomplementkod.

Variablerna sparas alltid efter Basicprogrammets sista rad enligt denna modell.

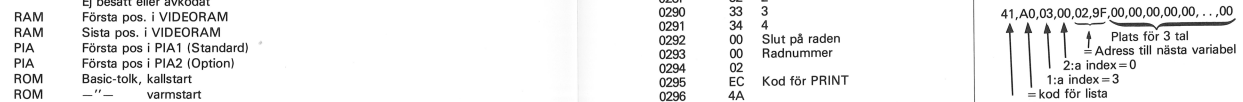
Som exempel visas nu hur olika variabler lagras.

1 Enkla variabler

A = 123

A¤ = "ABC"

DIM A(3)



2.4 Vad händer vid start?

När matningsspänningen slås till, får mikrodatorn en RESET-signal. Vad som händer sedan är följande:

1. Stackpekaren laddas med värdet 0030 Hex
2. De första 256 positionerna i minnet 0-ställs
3. PIA:n programmeras för att kunna läsa tangentbordet
4. Vissa pekare för operativsystemet initieras
5. Bildskärmen töms
6. Basictolken kallstartas
7. Den testar hur mycket fungerande minne som finns och nollställer detsamma
8. Initierar stackpekaren till den högsta minnespositionen
9. Initierar övriga pekare och konstanter.
10. Skriver ut Esselte 100 och en prompt till bildskärmen
11. Börjar vänta på inmatning från tangentbordet.

2.5 Avbrott

De övriga avbrott som kan hanteras av mikroprocessorn MC6802 behandlas på lite olika sätt.

Software Interrupt = SWI medför att programmet avbryts och att exekveringen fortsätter i Minimonitorn, där t ex registren kan studeras. Detta beskrives närmare i kapitel 9.

Non-Maskable Interrupt = NMI kan endast förekomma efter modifiering av moderkortet, men medför då hopp till Basictolkens varmstart, ett slags mjuk-RESET.

Interrupt Request = IRQ medför hopp till ett avbrottshanteringsprogram. Där testas först om användaren har definierat en egen avbrottsrutin, om inte testas om PIA2 är monterad. I så fall uppdateras den realtidsklocka som finns i operativsystemet.

Om ej heller detta är fallet, fortsätter exekveringen i Minimonitorn.

Användaren kan alltså via PIA1 och printerporten initiera ett avbrott.

Exempel: Man vill ibland kunna avbryta ett basicprogram, dels som ett slags nödstopp, dels för att till exempel utföra någon tidskrävande datainsamling.

För att få datorn att kunna ta emot en avbrottsförfrågan, kan man t ex programmera PIA-kretsen så att någon av signalingångarna i printerkontakten ger avbrott.

Programexempel:

10 POKE(51209,53) — Gör ingång A avbrottskänslig

Om nu ingång A läggs till jord, kommer ett avbrott att initieras, exekveringen avbryts och processorn går ut i minimonitorn. Vill man i stället komma tillbaka till Basic varmstart, får man tillfoga:

20 POKE(9,126) —— Sätt adress till Basic varmstart.
30 POKE(10,208)
40 POKE(11,3)
50 POKE(68,128) —— Låt avbrottet gå till 0009H.

2.6 Exekveringshastighet

Den tid det tar för tolken att utföra några vanliga instruktioner framgår av följande tabell:

Typ av instruktion	Exempel	Tid (ms)
Byta rad/hitta nästa rad	**10 A=2** **20 B=19**	3
Hitta nästa instruktion när den följer efter "kolon";	**10 A=2 : B=7**	4
Ovillkorligt hopp	**10 GOTO 20**	3
Tilldelning av variabel	**A=123**	3
FOR . . NEXT-loop		10
Addition	**A=2+3**	4
Multiplikation	**A=3*7**	10
Division	**A=6/7**	40
Trigonometriska funktioner	**A=SIN(35)**	250
Exponentiering	**A=EXP(2)**	400
Ut till maskinkodsrutiner	**T=TIME(2)**	3

Några kommentarer till tabellen kan kanske vara på sin plats.

Vad bör man tänka på, om man vill att program skall exekveras snabbt?

1. Använd inte kolon för att packa programinstruktioner.
2. Aritmetiska funktioner bör ej delas upp på flera rader.
3. Konstanter bör ges variabelnamn.
4. Ofta använda variabler bör definieras i början av programmet, för att de snabbt ska hittas.
5. Låt datorn utföra så få beräkningar som möjligt inuti loopar.
7. Använd * istället för ∧. Till exempel Y=X*X i stället för Y=X∧2.
8. Använd listor med beräknade värden för ofta återkommande funktionsvärden.
9. Maskinspråksprogrammera.

2.7 Att hushålla med minnet

Det finns 7 289 byte för lagring av program och variabler. Följande punkter kan vara värda att tänka på, då Du lyckats fylla upp maskinens minne med långa program eller stora datamängder.

1. Varje Basicrad tar minst 6 byte (två för radnummer, två för internkoder och två för radlängden) förutom själva Basicsatsen.
2. Varje flyttalsvariabel tar 8 byte.
3. En lista av flyttalsvariabler tar 6+6N byte, N är antalet dimensionerade element.
4. Varje sträng tar 2 byte förutom det antal tecken som angivits med STRING-kommandot (normalt = 32).
5. En lista av N st strängar med vardera S tecken tar S+S*N byte.

Några råd

Använd kolon och skriv flera instruktioner på varje rad.

Skriv ? eller ; i stället för PRINT och ! i stället för REM.

Skriv t ex IF A=5 THEN 100 i stället för IF A=5 THEN GOTO 100.

Använd listor i stället för enkla variabler, dimensionera bara så många element som behövs. En lista upptar annars alltid 10 element. På detta sätt kan upp till 1 200 variabler lagras.

Bestäm längsta strängläng med STRING kommandot. Om inget annat angivits antas alla strängar ha längden 32 tecken. Kom också ihåg att en odimensionerad lista av strängar alltid har 10 strängelement.

Data och även text kan lagras direkt i RAM med kommandot POKE och hämtas med PEEK. På detta sätt kan tal mellan 0 och 255 lagras till ett antal av 7 000.

Har man plats och tid skall man definitivt inte följa råden i avsnitt 2.6 och 2.7.

De kan göra programmen både svårlästa och svårrättade. Nej, spara inte på programrader och kommentarer!

2.8 Systemswitchar

För att göra operativsystemet flexibelt, finns hos detta ett antal alternativfunktioner att välja mellan. Genom att skriva i vissa minnesceller, de s k systemswitcharna, kan man välja mellan alternativen.

Oftast är det en ensam bit i en minnescell som har en viss funktion och för att påverka den och endast den, går man tillväga på följande sätt:

Sätt bit 0 i cell 64 och nollställ de övriga:

POKE(64,1)

Nollställ den och de övriga: **POKE(64,0)**

Sätt bit 0 till 1:

POKE(64,OR(PEEK(64),1))

Nollställ den:

POKE(64,AND(PEEK(64),255-1))

Invertera:

POKE(64,XOR(PEEK(64),1))

Följande statusceller finns:

Adress Hex	Adress Decimal	Bit	Funktion
43	67	0	Transparent bildskärm
43	67	1	Skiftlås bort från tangentbord
43	67	4	Släck markören
44	68	0	Break från serieport
44	68	1	Tidbas ändras till APULS och BPULS
44	68	3	GET-vippa, rör ej!
44	68	6	?#5 går till port 4 ut
44	68	7	Egen avbrottsrutin i 0009H
45	69	0	8 bitar i serieut
45	69	1	Auto LF efter CR
45	69	2	Eko till serieutgång
45	69	3	Slå av ekot från tangent till bildskärm
45	69	4	Alla kontrolltecken ut från tangentbordet
45	69	5	Ingen fördröjning efter CR vid serieut
45	69	7	Ingen 'scroll' av bildskärmen

Exempel på användning av systemswitchar finns i kapitel 5.

Kapitel 3

Esselte 100 som systemkomponent

3.1 Inledning

Som de flesta smådatorsystem består Esselte 100 av en processor med beräkningskapacitet, ett primärminne av RAM-typ, en bildskärm för utmatning av resultat och ett tangentbord för inhämtning av data och program. Som massminne används en kassettbandspelare.

Esselte 100 utmärks av följande:

I: PORT-systemet
II: Egendefinierade in- och utmatningsrutiner
III: En mycket flexibel uppbyggnad.

3.2 Portsystemet

Som kärna i Esselte 100-systemet ligger en Basic-tolk. Tolken kommunicerar med omvärlden via fyra kommunikationsvägar:

1. Pekare och stack
2. Primärminnet
3. Ut-PORTEN
4. In-PORTEN

1. Pekare och stack är specifika platser i primärminnet där tolken lagrar sina pekarevärden, slaskvariabler, återhoppsadresser mm. Tolken initierar dessa värden av sig själv, men användaren kan under programmet ändra dem.
2. I primärminnet lagras programmet som en följd av textsträngar, åtskilda av kontrolltecken och radnummer. Texten lagras enligt ASCII-standard, utom de av tolken igenkända kommandoorden vilka ersätts av interna koder (adresser). I anslutning till programmet lagrar tolken upp alla de variabler som används av programmet.
3. Ut-porten är egentligen en programsekvens, som ligger utanför tolken i operativsystemet. Detta underprogram får ta emot text från tolken för vidare befordran till användaren. I operativsystemet finns ett antal olika programsekvenser för utmatning av text till bildskärm, till printer och till bandspelare. De kallas PORT 1 , 2 resp 3.
4. In-portar är på samma sätt program som vidarebefordrar text från användaren till tolken. I operativsystemet finns program som hämtar text från tangentbordet, serieingången, PROM-kassett och från bandspelare.

I ett Basicprogram väljer programmeraren den PORT vilken utskrift skall ske till resp hämtas ifrån genom att ange portnumret:

Exempel:

PRINT#1, A¤	Skriv på bildskärm
LIST#2	Lista på printer
INPUT#3, A¤	Hämta från bandspelare

Normalt använder datorn automatiskt PORT 1, dvs den läser från tangentbordet och skriver på bildskärmen utan att porten behöver anges. Porten kan bytas till valfri port med ett kommando.

Exempel:

PORT=2	Seriein-/utgången

Vissa andra kommandon utför också portbyte, t ex kommandot PROM, som sätter PORT=6 för att läsa från en ansluten PROM-kassett. Kommandot MASTER sätter PORT=7. Detta är en serieport som används vid inkoppling till yttre dator eller skivminnesenhet.

Användaren kan om han vill skriva egna in- och/eller utmatningsrutiner. PORT=4 är ledigt för detta. Han kan då t ex anpassa utskriften till en yttre bildskärm/skrivare, ansluta paralleltangentbord eller hålremsläsare etc.

3.3 Möjligheter tack vare portsystemet

3.3.1 PROM-kassett (Promett)

I Esselte 100 finns ett kommando PROM, som gör att datorn börjar läsa in ett program från en programkassett. Programkassetten kan innehålla både Basic-program och maskinkodsrutiner, var för sig eller blandat. Ofta använda program kan "Prommas" så att ett programbibliotek kan byggas upp. Överföringen från kassetten sker seriellt/synkront via datorns printerkontakt. Hastigheten är mycket hög, 8K minne fylls på ett par sekunder! I PROM-kassetten kan även kommandon lagras, så att program kan startas automatiskt och även kan göras listskyddade. Då kommandot PROM ges överförs kommandot till en rutin som läser ur promkassetten istället för från tangentbordet. I denna finns lagrat kommandon, Basicprogram eller maskinkod. Möjlighet finns här att mata in små versioner av andra språk (<8K) t ex editor, assembler, PILOT etc.

En viktig funktion är också att periferenheter kan förses med styrprogram, som läses in då enheten anslutes. Det enda användaren gör är att ansluta enheten till printerkontakten och skriva PROM på tangentbordet.

3.3.2 Yttre terminal

Med kommandot PORT=2 kommer all in och utmatning att ske via seriein-/utgången. Denna sänder och tar emot data i serieform, 7 eller 8 databitar, med hastigheter mellan 75 och 9 600 baud. Allt är omställbart från programmet. En yttre terminal kan nu användas för all programmering och utmatning av resultat. T ex kan en 80-teckens bildskärmsterminal eller en skrivande terminal av typ Texas Silent användas. Anpassning av radlängd sker med LINE=xx kommandot.

3.3.3 MASTER

Esselte 100 kan kopplas samman med ett intelligent skivminne. Med kommandot MASTER kan en användare få tillgång till de data och program som finns lagrade på skivminnet. Han kan också lagra sina egna program och data. Kommandot MASTER testar först om förbindelsen är ledig, innan porten ändras.

Esselte 100 kan förstås också användas som terminal till ovanstående intelligenta skivminne, som alltså kan vara en större och mer kraftfull dator med större minne, andra språk mm. Antingen använder man terminalemuleringsprogrammet i Esselte 100 eller också kan Mastern överföra lämpligt program till Esselte 100.

Hastigheter:

Standardhastighet är vald till 300 baud. I de flesta fall då all in- och utmatning sker mjukvarustyrt, kan dock hastigheter och format ändras från programmet eller t o m från en yttre dator. Detta innebär att kontakt med yttre dator tas i 300 baud och då denna kontakt är uppnådd ökas automatiskt hastigheten till upp till 9 600 baud. Genom att föra över portrutiner kan man också få helt valfria format, synkrona eller asynkrona med hastigheter upp mot 40Kbits/s.

Själva portsystemet kan också användas för att bygga upp databaser och för att göra självskrivande program.

Kapitel 4

Bildskärmen och tangentbordet

4.1 Bildskärmen

Bildskärmen är den normala presentationsenheten till datorn. Med PRINT och PRINT#1 kommer datorn att på bildskärmen skriva ut önskad text. Texten på bildskärmen finns lagrad i ett RAM ("video-RAM") på 1024 byte, där varje byte motsvarar en teckenposition på bildskärmen. Med hjälp av ett 2048 byte EPROM ("tecken-PROM"), kommer den information som står i Video-RAM att omvandlas till ett lämpligt mönster på skärmen. De av den s k ASCII-koden definierade tecknen finns lagrade i detta tecken-PROM, tillsammans med ett antal andra grafiska symboler. Här skall endast behandlas bildskärmens funktion i samband med utskrift av text.

4.2 Markören

För att underlätta inmatning av text, finns en markör (eng cursor) på bildskärmen, placerad där nästa tecken kommer att skrivas ut. Tänk Dig gärna markören som typhuvudet på en skrivmaskin! Den består normalt av ett streck "_". Det är också möjligt att flytta markören utan att någon text skrivs ut. Detta sker med de olika "piltangenterna".

← Medför att markören backar samt att datorn suddar tecknen ur radbufferten.

CTRL← Markören backar (utan att sudda).

→ Markören flyttas framåt, det tecken markören stod på matas in/kopieras till radbufferten.

CTRL→ Markören flyttas framåt (utan att kopiera).

SHIFT← Markören flyttas ett steg uppåt.

SHIFT→ Markören flyttas nedåt ett steg.

Markören kan flyttas under programkontroll på följande sätt:

CTRL← = **PRINT CHR¤(8);**
CTRL→ = **PRINT CHR¤(26);**
SHIFT← = **PRINT CHR¤(11);**
SHIFT→ = **PRINT CHR¤(10);**

Med dessa funktioner kan t ex utskrifter formateras. Ibland vill man att markören ska släckas. Detta görs med CTRL-Ä.

Då bildskärmen har fyllts, kommer den översta raden att suddas och de övriga flyttas upp ett steg, skärmen "rullas" (eng scroll). Med kommandot CLR eller ?CHR¤(12) kommer skärmen att fyllas av blanktecken och markören att placeras i övre vänstra hörnet. Från tangentbordet åstadkommes denna funktion med kommandot CTRL-L.

4.3 Extra funktioner

I följande tabell har de kontrollkoder som styr bildskärmen samlats.

Från tangentbordet	I program	Funktion
CTRL-C	?CHR¤ (3)	Slår av funktioner, markerade*
CTRL-G	?CHR¤ (7)	Sänder ut en ton på A UT=BELL
CTRL-H	?CHR¤ (8)	Backar markören, suddar ut i radbufferten vid CTRL-H
CTRL-I	?CHR¤ (—)	Kopierar från bildskärmen till radbufferten
CTRL-J	?CHR¤ (10)	Flyttar markören nedåt=LINE FEED
CTRL-K	?CHR¤ (11)	Flyttar markören uppåt
CTRL-L	?CHR¤ (12)	Suddar sidan, markör till 1:a pos, =FORM FEED
CTRL-M	?CHR¤ (13)	Vagnretur, markör till början av rad=CARRIAGE RETURN
CTRL-N	?CHR¤ (14)	Suddar resten av raden
CTRL-O	?CHR¤ (15)	Suddar resten av sidan
CTRL-P*	?CHR¤ (16)	Slår av/till skiftlåset för gemener
CTRL-Å*	?CHR¤ (29)	Slår av markören
CTRL-^	?CHR¤ (30)	Bestämmer att här är rad 1 på sidan, allt ovanför skyddas vid scroll

*Slås av med CTRL-C

Till dessa kommer också några Basic-kommandon:

CLR	Tömmer skärmen, markör till övre vänstra hörnet
GCLR(A,B)	Tömmer skärmen från alla tecken med ASCII-koder mellan A och B
GSTO(M)	Lagrar bildskärmen i minne nr M (1–6)
GRCL(M)	Hämtar en bild ur minnet
PRINT TAB(X)	Flyttar in markören till kolumn X
P=POS	Ger variabeln P markörens kolumnposition
CUR(K,R)	Placerar markören på kolumn K och rad R (1–32)
MOVE(X,Y)	Placerar markören i koordinat X,Y (0–95, 0–127)
PRINT#5,	Skriver med förstorad text. Storlek väljs med SIZE
SIZE=0	Normal text, 32 tecken/rad
SIZE=1	Medelstor text, 10 tecken/rad
SIZE=2	Stor text, 5 tecken/rad

4.4 Tangentbordet

Tangentbordet består av en skrivmaskinsdel och en numerisk del, totalt 72 tangenter av "klicktyp". De 72 tangenterna är organiserade i en matris om 8×10 noder. Varje tangent kan kortsluta ett rad/kolumnpar. Datorn avsöker denna matris via en periferikrets, en PIA (MC6821). Då en nedtryckning avkänns, kommer motsvarande tangentkod (ASCII) att räknas ut med hjälp av rad/kolumnkoordinaterna.

Tangentbordet har tillsammans med bildskärmen portnummer 1 och är det naturliga sättet att mata in data och text till datorn.

4.5 Att hämta instruktioner från tangentbordet

Ett antal instruktioner finns för att inhämta information från tangentbordet:

INPUT#0,A	Hämtar ett enda tecken och väntar inte på RETURN
INPUT A¤	Hämta in tecken för tecken, till dess en vagnretur eller ett komma påträffas
LINPUT A¤	Hämta in alla tecken till dess vagnretur påträffas
A=KEY	Gör en enda avsökning av tangentbordet, om någon tangent är nedtryckt tilldelas A motsvarande teckens ASCII-värde. Är ingen tangent nedtryckt blir A=0.

Exempel:

Ett program skall kunna avbrytas genom att man trycker på en numerisk tangent. Programmet skall då fortsätta på en programrad som ges av tangenten ifråga.

```
10
20  GOSUB 100
30
.
.
.
100  !AVBROTT
110  K=KEY
120  IF K=0 THEN RETURN
130  K=K-48
140  ON K GOSUB 1000,2000,3000,4000,
     5000
150  RETURN
```

Program

Detta tillåter att man använder den numeriska delen av tangentbordet som användardefinierade funktionstangenter.

Ett annat exempel:

```
100  ON AND(KEY,15)+1 GOTO 110,1000,
     2000
```

Blir 1 om ingen tangent trycks ner
" 2 " tangent 1 trycks ner
" 3 " " 2 " "

KEY kan förstås också användas för att styra t ex markören i grafiska program. Man bör då släcka tangentbordsekot till bildskärmen, se 2.8.

Om man tillåter att programmet stannar upp för att invänta en nedtryckning, kan man istället använda:

```
110 INPUT#0,K
```


4.6 Gemener

Basictolken kräver att alla instruktioner skrivs med versaler (stora bokstäver). Tangentbordet är därför normalt försett med skiftlås, så att endast versaler kan skrivas. Skiftläset kan man ta bort med CTRL-P. Tangentbordet fungerar därefter som en vanlig skrivmaskin (versaler vid SHIFT). Teckengeneratorn kopplas samtidigt om för att kunna presentera gemener. Normalt används nämligen dessa ASCII-koder till vissa grafiska symboler. Detta innebär att det ibland kan se lite roligt ut vid listningar av text, skrivna med gemener, om inte CTRL-P eller ?CHR¤(16) först skrivits.

Återgång till skiftlås sker med ytterligare ett CTRL-P eller ev CTRL-C (?CHR¤(16) resp ?CHR¤(3)).

4.7 Att skydda fält på bildskärmen

Funktionen CTRL-^ är en mycket användbar funktion. Exempelvis kan en överskrift skrivas högst upp på skärmen varefter frågor och svar kan skrivas på skärmens nederdel utan att överdelen scrollar.

Exempel:

```
10 CLR
20 ?"ÖVERSKRIFT"
30 CUR(1,15)
40 ?CHR¤(30);
50 INPUT"NAMN",A¤
60 GOTO 50
```

Prova!

Vid listningar kan t ex de 15 första raderna listas (där dimensioneringar och kommentarer finns), CTRL-^ tryckas ned och resten listas på den nedre halvan av bildskärmen. Endast den nedre bildskärmshalvan scrollas.

4.8 Texthantering

Kombinationer av tecken kan sättas samman till en sträng, där tecken och kontrollkoder kan blandas.

Exempel:

Strängen A¤ skall innehålla två rader bestående av strängarna B¤ och C¤ rakt ovanför varandra.

```
10 B=LEN(B¤)
20 A¤=B¤
30 A¤=A¤+CHR¤(10)
40 FOR I=1 TO B
50 A¤=A¤+CHR¤(8)
60 NEXT I
70 A¤=A¤+C¤
80 CUR(5,5)
90 ?A¤
```

Bland de grafiska symbolerna finns bl a mycket tunna linjer för exempelvis linjedragning, tabellupprittning mm.

Exempel:

Strängen A¤ skall vara strängen B¤ men inramad av tunna linjer.

STRING=128

```
10  B¤="TEST"
20  B=LEN(B¤)
30  C¤=CHR¤(108)
40  D0=CHR¤(10)+CHR¤(8)+
    CHR¤(8)
50  FOR I=1 TO B
60  C¤=C¤+CHR¤(109)
70  D¤=D¤+CHR¤(8)
80  NEXT I
90  C¤=C¤+CHR¤(108)
100  A¤=C¤+D¤+CHR¤(110)+
     B¤+CHR¤(110)+D¤+C¤
110  ?A¤
```

Exempel på tabellritning:

```
0010 ! FUNKTIONSTABELL
0020 LINE=0:CLR:!INIT
0030 D¤=CHR¤(110):KOLUMN
0040 B¤=CHR¤(109)
0050 A¤=CHR¤(108)+B¤+B¤+B¤+B¤+
     B¤+B¤
0060 A¤=A¤+A¤+A¤+A¤
0070 A¤=A¤+A¤:!LINJE
0080 ! TABELLHUVUD
0090 ?"  X  1/X  X^2  X^3  SQR(X)"
0100 ? CHR¤(30);:!SKYDDA
0110 ! FUNKTIONER
0120 FOR X=0.1 TO 1.3 STEP 0.1
0130 R(1)=X:R(2)=1/X:R(3)=X*X:R
     (4)=X*X*X:R(5)=SQR(X)
0140 FOR I=1TO5
0150 R¤(I)=LEFT¤(STR¤(R(I))+"    ",5)
0160 NEXT I
0170 GOSUB 1000
0180 NEXT X
0190 END
1000 ! UTSKRIFT
1010 ? A¤;
1020 ? CHR¤(13);
1030 FOR I=1TO5
1040 ? D¤;R¤(I);
1050 NEXT I
1060 ? D¤
1070 RETURN
```

4.9 Att blanda gemener och grafik

För omställning till och från gemener används CHR¤(16).

```
A¤="L"+CHR¤(16)+"asse"
+CHR¤(16)+"B"
?A¤
```

ger Lasse B.

```
A¤=CHR¤(16)+"CHR¤(107)"+
CHR¤(16)+CHR¤(104)
PRINT 100; A¤
```

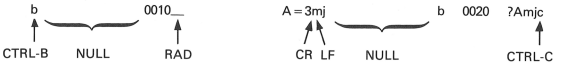
ger 100 kΩ.

4.10 Några speciella funktioner

Vill man att kontrolltecknen (t ex CR, CTRL-L eller LF) ska synas på skärmen, men inte utföras, kan man ställa om systemswitchen med POKE (67,1), se avsnitt 2.8. Man brukar säga att bildskärmen då blir transparent för kontrollmoder. Det här kan man utnyttja om man vill studera vad som spelats in på bandet då ett program lagrats.

Exempel:

```
10 A=3
20 ?A
```



Ytterligare möjligheter till styrning av bildskärm och tangentbord medges via systemswitcharna, se avsnitt 2.8.

Kapitel 5

Printer och terminal

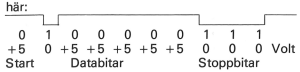
5.1 Format

Esselte 100 är utrustad med en serieutgång, kallad UT B. Kontakten sitter på sidopanelen och är märkt "PRINTER".

Utsignalen följer i huvudsak RS232 standard (TTL-nivåer, se nedan), och består av 1 startbit, 7 databitar och 2 stoppbitar.

En logisk nolla ut är lika med +5 Volt och en logisk etta lika med 0 Volt. Detta tolkas av många printrar som RS232-kompatibelt. (Egentligen specificeras RS232 av "0" = +12 Volt och "1" = −12 Volt.)

T ex bokstaven "A" (ASCII 01000001) matas ut så här:



5.2 Hastighet

Vid tillslag av Esselte 10 kommer serieutgången att ge hastigheten 300 baud. Efter vagnretur (CR) kommer en relativt lång fördröjning, detta för att långsamma printrar skall hinna utföra vagnreturen. Efter 32 tecken (utan vagnretur) ges en automatisk sådan, detta för att utskriften skall likna den på skärmen. Denna funktion är kopplad till LINE-kommandot, så att t ex LINE=40 ger 40 teckens utskrift och LINE=0 kopplar ur funktionen.

Om printern klarar högre hastigheter, kan utmatningshastigheten lätt ändras. Detta görs med kommandot TERM(bbbb). Möjliga hastigheter är 75, 110, 150, 300, 600, 1200, 2400, 4800, och 9600 baud.

Starta TERM-funktionen med lämplig hastighet och avsluta densamma direkt med ⟨CTRL-π⟩. Förutom att hastigheten ändras, kopplas också tidsfördröjningen ur.

Även från ett Basicprogram kan man ändra hastigheten. Detta görs genom att man skriver i de minnesceller där tidbasen finns lagrad, i adresserna 64 och 65.

För att få önskad hastighet skrivs POKE(64,A): POKE(65,B) där A och B fås ur följande tabell:

BAUD-rate	A	B
75	6	122
110	4	106
150	3	58
300	1	154
600	0	202
1200	0	98
2400	0	46
4800	0	20
9600	0	7

För att slippa tidsfördröjningen skriver man POKE(69,32). Den automatiska vagnreturen slipper man med LINE=0.

5.3 Optioner

Förutom ovanstående ändringar kan man också få 8 databitar, detta genom att skriva POKE(69,1) eller ev. POKE(69,0R(PEEK(69),1)). Om printern behöver extremt lång tid för återgång efter vagnretur, kan man ge ett långt LINE-kommando. T ex LINE=100 ger ca 3 sekunders fördröjning.

Någon egentlig handskakning finns inte och föreskrivs inte heller av RS232-standarden. Esselte 100 testar dock ingång A i printerkontakten innan den skickar ut data i en PRINT#2 sats. Om IN A=0 volt stoppas utmatningen. Genom att koppla en ev BUSY/READY signal från printern hit, kan man försäkra sig om att datorn ej skriver för fort till printern. Om inget är tillkopplat till IN A kommer den att ligga på +5 volt och datorn har fritt fram att skicka data. Om man skulle vilja sända data oberoende av IN A kan man använda PRINT#7, som ej testar densamma.

Denna BUSY/READY-funktion används bl a till en printer-multiplexer, som gör det möjligt för flera Esselte 100 att dela på en printer.

5.4 Exempel på printrar

5.4.1 Trendcom/Esselte

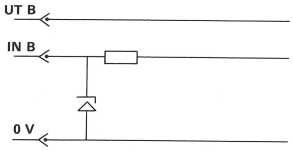
Detta är den printer datorn förväntar sig, utskrift med 32 teckens bredd, fördröjning vid vagnretur ca 1 sekund. Printern har plats för 40 tecken och dessa kan utnyttjas genom kommandot LINE=40. Om utskrift bara sker av en enda rad i taget, kan fördröjningen tas bort med POKE(69,32).

5.4.2 Texas Silent

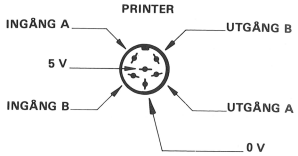
Ställ hastighetsomkopplaren i läge HI=300 baud. Radlängden ställs om med LINE=0. Listade rader blir nu max 77 tecken långa och vanliga PRINT-rader hur långa som helst. LINE=0 ger också en mindre fördröjning, ca 0,3 sekund. Vill man helt ta bort denna, skriver man POKE(69,32). Denna printer, som också är försedd med ett tangentbord, går utmärkt väl att använda som extra tangenbord till datorn. Den vill dock ha ett eko tillbaka, vilket åstadkommes med POKE(69,36). För att överföra kommandot till Silent-terminalen, ges kommandot PORT=2. Utsignalen från Silent-terminalen måste dämpas till TTL-nivåer, se 5.5, och matas in på IN B på printerkontakten.

5.5 Interface-kontakten

För att underlätta sammankoppling av Esselte 100 med andra printrar och terminaler, ges här en enkel ritning på ett interface för RS232. Inget har gjorts för att höja nivån från Esselte 100, däremot skyddas den för överspänningar.



Om man vill, kan en högimpediv hörtelefon kopplas mellan UT A och jord (helst en piezoelektrisk). I denna kommer att höras en ton då BELL funktionen används, antingen med ⟨CTRL-G⟩ från tangentbordet eller med PRINT CHR¤(7) i ett program. APULS:OUT(X,Y) kommer också att höras här.



5.6 Esselte 100 som terminal

5.6.1 TERM

Med hjälp av kommandot TERM(xxxx), kommer Esselte 100 att fungera som en bildskärmsterminal. Kommandot kan ges direkt från tangentbordet eller från ett program. Talet xxxx anger den hastighet som överföringen skall ske med. xxxx = 75, 110, 150, 300, 600, 1200, 2400, 4800 eller 9600 baud.

Då kommandot skrivits och RETURN-knappen tryckts ned kommer ett terminalemuleringsprogram att startas. Tangentbordet avsökes och då en tangent trycks ned, kommer motsvarande ASCII-kod att sändas ut på serieutgång B. Då ett meddelande avslutats (med RETURN), börjar programmet att lyssna på serieingång B. Alla inkommande tecken kommer att skrivas på bildskärmen. För att sända igen finns två olika sätt, beroende på aktuell överföringshastighet:

1) xxxx = 75, 110, 300 eller 600 baud.

Tryck bara på önskad tangent, mottagningen avbryts och den nedtryckta tangentens kod sänds. Observera att terminalen inte lyssnar förrän sändningen avslutas.

2) xxxx = >600 baud samt 150 baud.

Tryck en gång på CTRL-tangenten och fortsätt sedan med önskade tangenter.

Att proceduren är olika för olika hastigheter, beror på att datorn inte hinner känna av alla tangenter hur fort som helst. För att avsluta TERM-programmet skriver man bara CTRL-π. I TERM-mod fungerar de vanliga kontrollkoderna, så att t ex CTRL-P ger små bokstäver. Markörförflyttningar och kopieringstangenten fungerar som vanligt.

5.6.2 Formatändring

Alla funktioner, små bokstäver, scroll, automatisk radframmatning, släckning av markör etc. fungerar även i TERM-mod och ställs om som vanligt med systemswitcharna enligt avsnitt 2.8.

5.6.3 Några tips

Den mottagande datorn kan råka ge ett eko tillbaka till Esselte 100, slå av detta! Om det går att justera radlängd, välj 32 tecken/rad. Om det går att få fördröjning efter vagnretur, gör detta. Se till att antalet databitar är lika och att inte datorn ifråga sänder och vill ha paritetsbitar. Vissa datorer kan fås att stoppa utskrift med CTRL-S och starta utskrift med CTRL-Q. Det går att skicka CTRL-tecken från Esselte 100 i alla hastigheter, men observera att alla kontrolltecknen inte sänds ut utan modifiering, se 2.8. Det är också möjligt att få Esselte 100 att sända ett extra tecken efter carriage return, t ex då CTRL-Q. Normalt sänder den alltid ett nulltecken (0) efter CR, vill man ändra något annat skriver man in detta tecken i cell 004D hex (77dec). Vill man stoppa nulltecknet skriver man in talet 128 i denna cell. Normalt sändes 7-bitars ASCII-kod med 1 start- och 2 stoppbitar.

Efter CR sänds ett null-tecken. Följande kontrolltecken sänds: A, B, C, D, E, F, G, H, J, M, X. Om man sätter bit 4 i cell 69 (POKE(69,16)) innan TERM får man ut alla kontrolltecknen A—Ö. Då fungerar dock inte editeringsfunktionen så bra. Vill man ha t ex LF efter CR skriver man in dess ASCII-kod i cell 77_{10}, i detta fall POKE(77,10). Normalt lyssnar Esselte 100 inte förrän efter CR. I vissa fall är detta olämpligt. Genom att skriva talet 0 i cell 78, kommer Esselte 100 att lyssna direkt efter varje tangent. Lämpligt är då att ta bort det nulltecken som sänds ut med ett POKE(77,128). I denna funktion blir Esselte 100 mycket störningskänslig och "skräptecken" kan synas på bildskärmen.

Man kan i detta läge vilja att endast mottagna tecken skall synas på bildskärmen. Släck då bildskärmsekot från tangentbordet med POKE(69,8) innan TERM-kommandot. Esselte 100 scrollar normalt inte bildskärmen i TERM-läge. Detta beror på att scroll tar för lång tid, istället fortsätter utskriften högst upp på skärmen. Om överföringshastigheten är låg, kan det vara snyggt att ha scroll, gå då ur TERM, nollställ bit 7 i adress 69 och gå tillbaka till TERM(0)!

Nollställning: POKE(69,AND(PEEK(69,127)).

Som synes finns många möjligheter att ändra terminalfunktionen. Till sist kan nämnas att man mycket enkelt själv kan modifiera denna funktion, rutiner för serieut/in, tangentbord, bildskärm mm finns tillgängliga.

5.6.4 Några exempel

1) 300 baud, 7 bitar

```
skriv    TERM(300)      ⟨RETURN⟩
skriv    Meddelande     ⟨RETURN⟩
lyssna
skriv    Meddelande     ⟨RETURN⟩
avsluta  CTRL-π
```

2) 300 baud, 8 bitar

```
skriv    POKE(69,1)
skriv    TERM(300)
```

som ovan!

3) 300 baud, 7 bitar
automatisk radframmatning

```
skriv    TERM(300)
         POKE(69,2)
         TERM(300)
```

4) Gör Esselte 100 till en terminal. 600 baud, 2 stoppbitar, 8 databitar, små och stora bokstäver, scroll. CTRL-Q efter varje RETURN.

```
skriv
TERM(600)
CTRL-π
POKE(69,49)   1 bitar + CTRL + fördröjning
                8    16    32    = 49
POKE(77,23)
TERM(Ø)
```

Kapitel 6

Sammankoppling av flera Esselte 100

6.1 Esselte 100 som terminal till en annan Esselte 100

Koppla ihop de två datorerna med en terminalkabel till PRINTER-kontakterna. Kabeln ska vara kopplad så att Bin på den ena datorn kopplas till But på den andra och vice versa. Dessutom ska datorerna ha gemensam jord. Se först till att båda datorerna använder samma överföringshastighet. Normalt är hastigheten 300 baud om ingen åtgärd vidtagits för omställning av hastigheten (se 6.2 nedan). Skriv kommandot PORT=2 (följt av RETURN) på "slaven". Den andra, "mastern", ställs i terminalmod med kommandot TERM(300), följt av RETURN. Om man vill, kan kommandot LINE=0 ges först, vilket har till följd att överföringen går något snabbare, se nedan.

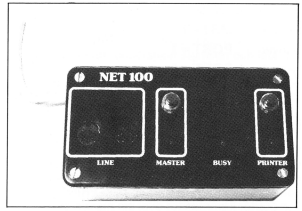
Mastern kan nu köra slaven, lista, ändra osv. Han kan också skicka meddelanden till slaven med ?#1,"xxxxxxx". Normalt blir överföringen ganska långsam, eftersom slavdatorn ger en lång tidsfördröjning efter varje RETURN. Detta har gjorts för att möjliggöra användningen av långsamma printrar. Mastern kan snabba upp överföringen med kommandot LINE=0, som tar bort denna extra tidsfördröjning.

Observera att mastern inte har möjlighet att stanna program i slaven med CONTROL-C, om inte programmet innehåller INPUT-satser. Detta beror på att den rutin som testar för avbrott, endast avsöker slavens tangentbord. Serieporten, som används vid master/slav-konfigurationen, kan dock fås att ingå i avbrottsrutinen, genom att kommandot POKE(68,1) ges. Vilken tangent som helst på mastern avbryter då programmet i slaven. Kom ihåg att ställa tillbaka funktionen med POKE(68,0) då maskinen inte längre ska köras som slav!

6.2 Lärarslinga

6.2.1 Hårdvara

Flera Esselte 100 går att koppla samman i en slinga, så att de var för sig kan kommunicera med en lärardator, "master". Vidare kan godtyckligt många Esselte 100 dela på en gemensam printer. För detta åtgår en enkel kopplingsbox vid varje dator. Kopplingsboxarna förbinds med varandra via skärmad fyrledare. Den totala ledningslängden bör inte överskrida 30 meter. På kopplingsboxarna finns tre lysdiodindikatorer och två tryckknappar, för statusindikering och modval. Två moder finns, MASTER och PRINTER. Ena lysdioden indikerar att MASTER-mod valts, den andra PRINTER-mod och den tredje visar om slingan är upptagen av trafik.



6.2.2. Användning

Om en elev vill lista program, eller köra ett program som ger utskrift på printern, ska hon till att börja med se efter att slingan är ledig. Slingan är ledig om "UPPTAGET"-indikatorn är släckt. Sedan väljs PRINTER-mod genom att tryckknappen "PRINTER" trycks ned en gång. Härvid tänds både "PRINTER"- och "UPPTAGET"-indikatorerna. Printern kan nu disponeras godtyckligt länge, precis som vanligt, dvs program kan listas med LIST#2 och utskrift erhålls med PRINT#2. Ingen annan kan nu komma åt printern. När printerkörningen avslutats, trycks tryckknappen "PRINTER" ned en gång till, varvid slingan åter blir ledig och indikatorerna slocknar. Tryckknappen har alltså en växlande funktion, vartannat tryck kopplar in datorn till printern, vartannat kopplar bort den.

MASTER-mod medger kommunikation med läraren, "mastern". Läraren kan då lista, ändra och köra program i elevens maskin. Han kan också skicka meddelanden till eleven och till och med överföra elevens program till sin egen maskin. Master-datorn ska från början stå i terminalmod, 300 baud. När en elev av någon orsak vill ha sitt program studerat eller åtgärdat av läraren, kontrollerar hon först att slingan är ledig, dvs att "UPPTAGET"-indikatorn är släckt. Därefter väljs "MASTER"-mod med tryckknappen "MASTER", varvid både "MASTER"- och "UPPTAGET"-indikatorerna tänds. Eleven skriver sedan kommandot MASTER (följt av RETURN), varvid läraren får kommandot över elevdatorn. När läraren vidtagit lämpliga åtgärder, t ex listat och ändrat i programmet, återfår eleven kontrollen över sin dator genom att läraren skriver PORT=1. När kommunikationen är avslutad, trycker eleven ned tryckknappen "MASTER" en gång till, varvid slingan åter blir ledig. Ett litet exempel kan kanske vara på sin plats:

Läraren		Eleven	
Skriver	**LINE=0** **TERM(300)** ⟨**CR**⟩ ⟨**CR**⟩		
lyssnar nu!			
		Skriver	**MASTER** ⟨**CR**⟩
Ser nu	#		
Skriver	**LINE=0** ⟨**CR**⟩		
Skriver	**?#1,"OK!"**	Ser nu	**OK!**
Skriver	**LIST**		
Ser nu programmet			
Ser felet och rättar			
Skriver	**LIST#1**	Ser nu programmet	
Skriver	**PORT=1**	Ser	**FEL I RAD 10**
Skriver	**PORT=1**	Återfår kommandot.	

6.2.3 Några kommentarer

Kom ihåg att mastern lyssnar bara efter det att dess RETURN-tangent tryckts ned! Läraren har möjlighet att kommunicera med elevens bildskärm genom att styra utskriften dit med PRINT#1. Eleven återfår kommandot över sin maskin genom att läraren skriver PORT=1. Tänk på att läraren verkligen kör elevens maskin via sin egen terminal! Observera också att elevens integritet aldrig kan kränkas, ingen kan titta på hans program eller ändra i programmen om han inte själv vill!

Vad händer egentligen om en elev börjar printa ut program till printern eller skriva MASTER, trots att slingan är upptagen av någon annan? Ja, som väl är händer just ingenting! Ingen annan kan störa pågående trafik på slingan, eftersom Esselte 100 och kopplingsboxen är försedda med kretsar som spärrar utmatning då slingan är upptagen.

6.2.4 Ökning av överföringshastigheten

Överföringshastigheten kan ökas till betydligt högre hastighet än 300 baud, speciellt vid korta slingor. Information om hastigheten finns lagrad i två minnesceller i datorn, vid ändring fordras följande procedur: Förutsätt att uppstartning har skett enligt exemplet i 6.2.2. Båda datorerna går då i 300 baud. Antag att läraren vill höja hastigheten till 600 baud. Han tillfogar då två programrader i elevens maskin.

```
1 POKE(64,0):POKE(65,202)
2 STOP
```

varefter han startar prorgrammet med RUN. Elevens maskin byter nu hastighet till 600 baud. På lärarens bildskärm syns nu diverse "skräptecken", eftersom elevmaskinen skriver ett #-tecken i 600 baud, något som lärarmaskinen inte kan ta emot. Läraren måste nu gå ur terminalemuleringsprogrammet. Detta görs med CONTROL-π. Han skriver sedan TERM(600), följt av RETURN, och har därefter kontakt i 600 baud. Därefter kan raderna 1 och 2 raderas bort. Om så önskas kan ännu högre hastigheter väljas, t ex POKE(65,20) ger 4800 baud. Tänk dock på att normala printrar är avsedda för 300 baud!

6.2.5 Överföring av program

Ibland vill läraren kopiera ett program från en elev. Detta kan han göra om han tillfogar följande överföringsrutin till elevens program:

```
1 WAIT(10000)
2 LIST#7          listar till serieutgången
3 ?#7,"PORT=1"    återställer lärarmaskinen
4 PORT=1          återställer elevmaskinen
5 STOP
```

Läraren tillfogar dessa fem rader i TERM-mod och startar sedan programmet med RUN. Läraren har därefter 10 sekunder på sig att gå ur TERM-moden med CONTROL-π och skriva PORT=2.

Efter de tio sekunderna börjar elevmaskinen sända över en listning till lärarens dator. Då listningen är slut sänds kommandot PORT=1, så att läraren återfår kommandot över sin dator. Slutligen återfår eleven kommandot över sin dator. Programmet finns nu lagrat i lärardatorn och kan listas och köras som vanligt. De tillfogade raderna 1—5 tas lämpligen bort före körning!

Samma procedur kan även användas åt andra hållet, dvs läraren kan sända ut färdiga program till eleverna. Vidare finns en handskakningsrutin att tillgå för sådana applikationer, som fordrar extremt höga hastigheter.

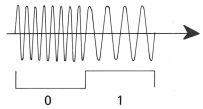
Kapitel 7

Bandspelaren

7.1 Format

Ett massminne är ett minne, i vilket datorn kan lagra större mängder data och program under godtyckligt lång tid. Esselte 100 kan utnyttja en vanlig kassettbandspelare som massminne.

Data spelas in på bandet som kombinationer av två toner. Formatet är välkänt inom smådatorindustrin och kallas Kansas City standard. Data matas ut seriellt i ett asynkront format med 1 startbit, 8 databitar och 2 stoppbitar. En nolla motsvaras av 4 perioder av en 1200 Hz sinuston och en etta av 8 perioder av en 2400 Hz ton.



Ett program i operativsystemet genererar dessa tonsignaler, de filtreras och dämpas innan de matas ut på den 5-poliga DIN-kontakten, märkt "TAPE".

Vid inmatning av data från band, går signalen via en komparator, där signalen omvandlas till de logiska nivåer mikroprocessorn arbetar med, 0 och 5 volt. Ett program undersöker signalen och letar reda på ettor och nollor i signalen. Överföringshastigheten till och från kassettband blir ungefär 30 tecken/sek, vilket inte är speciellt snabbt. Signalamplituden från bandspelaren ska vara mellan 0,5 V och 1 V. Datorns inimpedans är 10 kΩ på TAPE-kontakten.

Den långsamma hastigheten gör överföringen mycket säker, även med en enklare banspelare.

7.2 Vad sänds ut?

Med kommandot SAVE startas ett program som listar ut programmet till band. Före varje rad sänds ett CTRL-B och sedan radnummer. Efter varje rad sänds ett CR samt ett antal Null-tecken. Dessa behövs för att datorn skall hinna läsa in och lagra en rad innan nästa kommer. Om raderna inte ligger i nummerordning måste datorn få tid att arrangera raderna i stigande radnummerföljd.

Antal null är normalt 12, men kan av användaren ändras till 4 om man vill spara tid vid lagring och hämtning av program där raderna ligger i rätt ordning. Detta sker genom att man skriver önskat antal i adress 78, t ex POKE(78,4). Då hela programmet har lagrats, skickas dessutom ett CTRL-C ut till band för att markera slut.

Vid inhämtning av program listas programmet på skärmen. Tecknen kommer dock så tätt, att skärmen inte hinner rullas, utan markören går direkt upp till första raden igen. Då hela programmet lästs in töms skärmen.

7.3 Att lagra annat än program

Bandspelaren har portnummer 3 och det är teoretiskt möjligt att skriva direkt till band med PRINT#3 och även läsa med INPUT#3. Nu är det så, att utmatningsprogrammet inte matar ut någon ton då man inte anropar det. Detta innebär, att mellan utskrivna strängar kommer en massa "skräp" att lagras. Metoden är dock fullt användbar och det går till och med att skriva ut kommandon som RUN, LIST etc på bandet och låta datorn läsa dessa med INPUT#3. I kapitel 10 ska vi titta närmare på hur data kan lagras på band.

7.4 Att styra bandspelaren

Ingen styrning finns för bandspelaren, men det är ganska enkelt att använda en ledig signal i printerkontakten till detta. Se kapitel 11.

Kapitel 8

Grafik

8.1 Grafik på bildskärmen

Grafikhanteringen hos Esselte 100 skiljer sig något från de flesta andra små persondatorer. Normalt kostar hög grafisk upplösning mycket RAM-minne, som ofta tas ur datorns eget beräkningsminne. På Esselte 100 har valts en annan väg: Bildminnet är på 1024 byte, där varje byte representerar en position på bildskärmen. I teckengeneratorn finns 256 st olika tecken. Hälften av dessa är de vanliga gemena och versala bokstäverna samt diverse grafiska symboler, på samma sätt som i PET, ABC80 m fl.

De övriga tecknen bildar tillsammans ett undersystem av ett grafiskt punktmönster om 3×4 punkter. Om alla kombinationer skulle finnas med skulle det behövas $2^{12} = 2048$ tecken. Ur dessa har de 125 viktigaste valts ut. Tanken bakom detta system är att ett PLOT-kommando oftast används för att plotta/rita funktionskurvor. De viktigaste funktioerna, t ex Y = A*X, Y = SIN(X), karakteriseras av att för varje X-värde finns ett enda Y-värde! Systemet av plotpunkter är så utvalt att för varje horisontell position ska minst en punkt kunna placeras på valfri höjd.

Den som plottar en enda kurva på skärmen kommer aldrig att märka att vissa kombinationer saknas. Det är emellertid inte alltid möjligt att plotta två punkter omedelbart ovanför varandra. Praktiska prov har visat att man inget märker, även då ett flertal funktioner plottas samtidigt.

För att medge hög vertikal upplösning finns 7 st olika höga staplar, som tillsammans med BAR-kommandot ger en reell vertikal upplösning på 1/224.

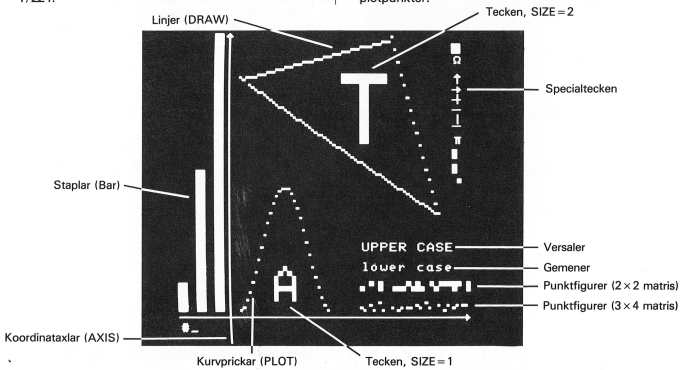
Vad vinner man nu med detta system?

- En synbarlig upplösning vid plottning av funktioner på 96*128 punkter.
- Normal "dotfunktion" (ABC80) om 64*64 punkter.
- Stapelframställning med upplösning 32*224.
- Ett antal grafiska symboler (PET).
- Alla tecken kan finnas samtidigt på skärmen, ingen omkoppling till grafisk mod erfordras!

8.1.1 Speciella grafikkommandon

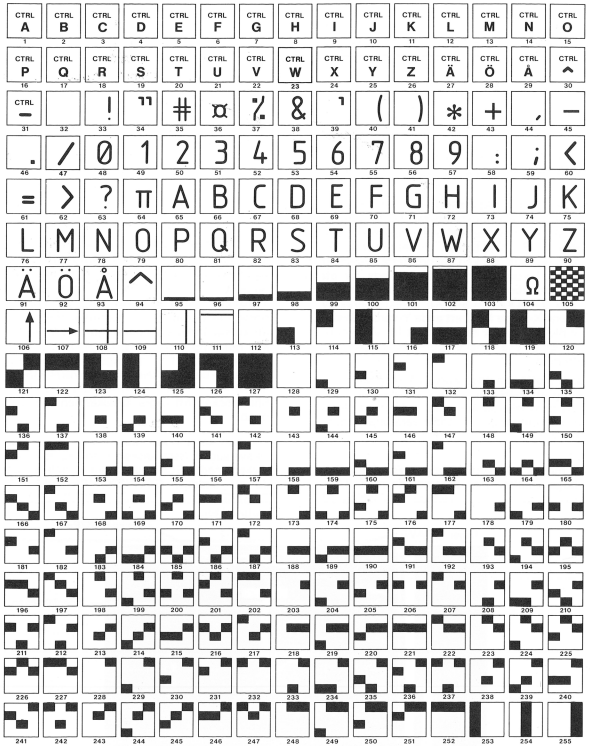
Möjlighet finns att lagra bildskärmens innehåll i RAM-minnet med kommandot GSTO(M). M är ett tal mellan 1 och 6 som anger en adress i minnet. Tillbakahämtning sker med GRCL(M). Dessa funktioner medger att kurvor kan ritas och sparas, för att kunna återkallas senare. En mall för en utskrift kan ritas upp och sparas, för att kunna användas flera gånger. Återhämtningen sker med en frekvens på ca 20 bilder/sekund, vilket innebär att animering av enkla rörelser kan ske. Möjlighet finns också att lagra bildskärmens innehåll på band.

Förutom kommandot CLR finns möjlighet att selektivt sudda vissa tecken med GCLR(A,B). Alla tecken med ASCII-kod mellan A och B kommer då att suddas. GCLR(106,110) suddar AXIS, GRCL(65,95) alla versaler, GCLR(128,254) alla plotpunkter.

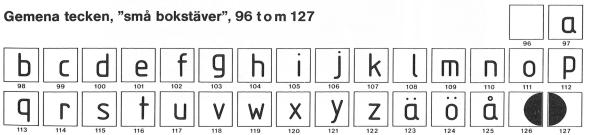


8.2 Grafiska tecken

I bildskärmens 1024 positioner, kan olika tecken placeras. Totalt finns 256 olika tecken lagrade i ett speciellt tecken-PROM, se nedanstående teckentabell:

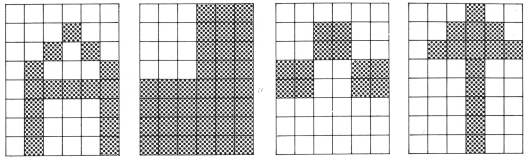


Gemena tecken, "små bokstäver", 96 t o m 127



Varje tecken är uppbyggt av punkter i en 6 × 8-matris.

Exempel:



Teckentyp:	Alfanumeriska tecken	Punktfigurer	Kurvprickar	Symboler
Användning:	Text	Figurer	Kurvritning	Div
Upplösning:	32 * 32	64 * 64	96 * 128	
Exempel på kommandon:	**PRINT"A"** **?"A"** **;"A"** **CLR** **CUR(A,B)** **TAB(A)** **POS** **LINE = K**	**?CHR¤(125)** **?CHR¤(127)** **SIZE** **PRINT#5,"A"** **CUR(A,B)** **TAB(A)** **SIZE = 1**	**CLR** **PLOT(X,Y)** **DRAW(X,Y)** **SEC(P)** **MOVE(X,Y)**	**?CHR¤(106)** **BAR(K,H)** **AXIS(X,Y)**

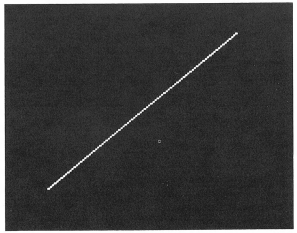
8.3 Att rita kurvor

PLOT-instruktionen gör kurvritning mycket enkel. PLOT(X,Y) sätter (om möjligt) en punkt i koordinaten (X,Y) $0 \leq X \leq 95$, $0 \leq Y \leq 127$.

Punkten (0,0) ligger i nedre vänstra hörnet, x-axeln är positiv åt höger och y-axeln är positiv uppåt.

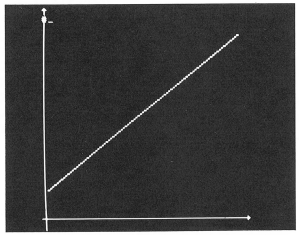
Exempel: Rita funktionen $Y = X + 12$ för $0 < X < 90$.

```
10 CLR
20 FOR X = 0 TO 90
30 Y = X + 12
40 PLOT (X,Y)
50 NEXT X
```



Exempel: Lägg till axlar

```
60 AXIS (0,0)
```



Om däremot t ex funktionen $Y = X^2$ ska ritas i intervallet $0 < X < 90$, kommer Y att bli för stor och ett felmeddelande erhålls. Följaktligen måste funktionen skalas, dvs anpassas till det tillgängliga koordinatsystemet, före uppritning.

```
 5 L1=127/(90*90)
10 CLR
20 FOR X=0 TO 90
30 Y=X*X
40 PLOT (X,Y*L1)
50 NEXT X
```

Ett mera generellt exempel (funktionen är Y = SIN(X)/X):

```
 10  CLR
 20  INPUT "XMIN,XMAX,YMIN,YMAX,
     X0,X1,Y0,Y1
 30  K1=95/(X1-X0):K0=-X0*K1
 40  L1=127/(Y1-Y0):LO=-Y0*L1
 50  FOR X=X0 TO X1 STEP 1/K1
 60  Y=SIN(X)/X
 70  IF Y<Y1 IFY>Y0 THEN 90
 80  GOTO 100
 90  PLOT (X*K1+K0, Y*L1+L0)
100  NEXT X
```

Prova t ex med XMIN = −12, XMAX = 12, YMIN = −0,5 och YMAX = 1.

Vill man illustrera tidsberoende funktioner grafiskt, är funktionen BAR(X,Y) mycket lämplig. I följande program används BAR(X,Y) för att generera en stapel vars höjd är ett mått på tiden.

```
 10  !REAKTIONSTID
 20  ?"NÄR EN BOKSTAV DYKER UPP"
 30  ?"TRYCKER DU PÅ
     MOTSVARANDE TANGENT"
 40  INPUT "TRYCK PÅ RETURN", A¤
 50  CLR
 60  FOR I=1 TO 20 STEP 3
 70  WAIT (RND*5000)
 80  CUR(I,29)
 90  A=INT (RND*28+65)
100  ?CHR¤(A)
110  CUR(1,1):P=200
120  FOR T=0 TO 200
130  BAR(I,T)
140  IF KEY=A THEN P=T: T=200
150  NEXT T
160  CUR(29,I)
170  ?P;
180  NEXT I
```

8.4 Att skriva med stora tecken

Esselte 100 har en unik egenskap, den medger utskrift av stora tecken på bildskärmen på ett mycket enkelt sätt. Med kommandot SIZE = N, där N = 0, 1 eller 2, bestäms teckenstorleken. För att skriva stora tecken används instruktionen PRINT#5, istället för PRINT.

Exempel:

```
10 SIZE=1
20 PRINT#5, "12345"
```

skriver 12345 på bildskärmen med 4× normal storlek. SIZE = 2 ger 8× normal storlek och SIZE = 0 ger normal storlek. Om man inte specificerar storleken med SIZE, förutsätter datorn normal storlek.

Eftersom Esselte 100 bl a är avsedd att användas som mätdator, kan de stora tecken med fördel användas för presentation av mätvärden för ett större auditorium. För den skull kan, förutom versaler, siffror och skiljetecken, tecknen m, μ, z, Ω och ° skrivas med stora bokstäver.

Exempel:

```
10 SIZE=1
20 ?#5, "I=";10;CHR¤(97);"A"
30 ?#5, "T=";20;CHR¤(39)"C"
40 ?#5, "R=";2.7;"K";CHR¤(99)
50 ?#5, "F=";100;"H";CHR¤(98)
```

Skriver: I = 10 μA
T = 20°C
R = 2.7 kΩ
F = 100 Hz

CHR¤(96) = m
CHR¤(97) = μ
CHR¤(98) = z
CHR¤(99) = Ω
CHR¤(39) = °

I stället för CHR¤(39) kan tecknet ' användas. SIZE = 1 medger utskrift av maximalt 10 tecken per rad, SIZE = 2 5 tecken per rad. För att texten ska avdelas riktigt, är det lämpligt att specificera radlängden 10 resp 5 med kommandot LINE.

8.5 Att skriva med STORA EGNA TECKEN!

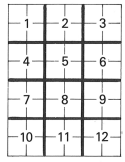
Om man skriver ?#5,CHR¤(X), där 100 ≤ X ≤ 109 skrivs ett eget definierat tecken på bildskärmen. Upp till 10 egna tecken kan definieras.

Definitionen görs i RAM, i adress 0000 och framåt.

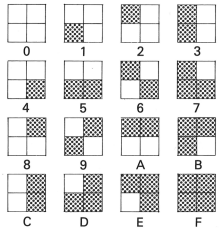
Varje tecken upptar 6 byte, dvs:

0000 − 0005 = CHR¤(100)
0006 − 000B = CHR¤(101)
osv.

Tecknet består av en 6 × 8 matris enl:

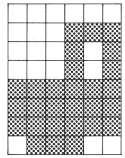


var och en av rutorna 1 − 12 kan tilldelas ett hexadecimalt värde 0 − F enligt nedan:



Dessa delmängder lagras 2 och 2 i konsekutiva RAM-adresser.

Exempel: Önskad figur:



1 = 0 8 = F
2 = 4 9 = F
3 = 5 10 = E
4 = 0 11 = F
5 = C 12 = A
6 = C
7 = F

Skriv nu in i minnet:

```
PATCH ⟨CR⟩
M  0000
0000  04
0001  50
0002  CC
0003  FF
0004  FE
0005  FA
⟨CR⟩
B                    tillbaka till BASIC.
```

Sätt SIZE = 1 och skriv ?#5,CHR¤(100);. Tillverka ytterligare ett tecken:

Lägg in:
```
0006  04
0007  55
0008  57
0009  FF
000A  FE
000B  AB
```

och skriv ?#5,CHR¤(100);CHR¤(101);

Detta går förstås att tillordna en variabel:

```
A¤=CHR¤(100)+CHR¤(101);
```

Följande program utnyttjar just den här figuren (ett ånglokomotiv!) och rör den i sakta mak över skärmen. Raderna 10 − 40 sköter om inskrivningen i RAM-minnet, data hämtas från raderna 240 och 250.

```
 10  FOR I=0 TO 11
 20  READ A
 30  POKE (I,A)
 40  NEXT I
 50  SIZE=1
 60  ?CHR¤(29)
 70  FOR R=4 TO 1 STEP -1
 80  READ B¤(R)
 90  NEXT R
100  CLR
110  A¤=CHR¤(100)+CHR¤(101)
120  FOR I=1 TO 25
130  CUR(I,10)
140  ?#5," ";A¤;
150  GOSUB 210
160  CUR(I+8, R+2)
170  ?B¤(R)
180  WAIT (200)
190  NEXT I
200  GOTO 100
210  R=R+1
220  IF R>4 THEN R=1
230  RETURN
240  DATA 4, 80, 204, 255, 254, 250
250  DATA 4, 85, 87, 255, 254, 171
160  DATA F, F, U, T
```

8.6 Rörlig grafik

Föregående programexempel genererade en rörlig bild. Vi ska här titta på några olika metoder för att åstadkomma rörliga bilder.

Exempel:

Partikel i en cirkel. Partikelns koordinater beskrivs av:

X = R * COS(F) + X0
Y = R * SIN(F) + Y0

Vi ska jämföra några olika metoder.

1. Plottning av partikelns rörelse.

```
10 CLR
20 R=20
30 FOR F=0 TO 2*π STEP π/16
40 X=R*COS(F)+48
50 Y=R*SIN(F)+63
60 PLOT (X,Y)
70 NEXT F
```

Det här programmet tar ganska lång tid att plotta, eftersom funktionsvärdet måste beräknas under rörelsens gång.

2. Beräkna först, plotta sedan.

```
  5 DIM C(32), S(32)
 10 CLR
 20 R=20
 30 FOR I=1 TO 32
 40 V=(I-1)*π/16
 50 C(I)=R*COS(V)+48
 60 S(I)=R*SIN(V)+63
 70 NEXT I
 80 FOR F=1 TO 32
 90 PLOT (C(F),S(F))
100 NEXT F
```

Det tar ganska lång tid innan plottningen påbörjas, men plottningen går fort.

I exemplen ovan lämnar partikeln spår efter sig på skärmen. Vill man inte ha några spår, kan föregående plotpunkt raderas med instruktionerna MOVE och ?" ".

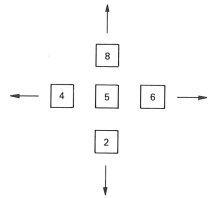
Vid alla rörliga bilder är det lämpligt att ha markören släckt, ?CHR¤(29).

Om man har en bakgrund som rörelsen sker över, är det inte så lämpligt att radera med MOVE, eftersom då även bakgrunden raderas. Med hjälp av GSTOP och GRCL kan då bakgrunden sparas.

1. Rita bakgrund
2. Lagra (GSTO)
3. Rita den rörliga figuren
4. Hämta tillbaka bakgrunden (GRCL)
5. Beräkna nytt läge på den rörliga figuren
6. Gå tillbaka till punkt 3

Exempel:

Spelet "Asteroid", där man på viss tid ska skjuta ner så många asteroider som möjligt. Tid väljs, sedan styrs projektilen med numeriska tangentbordet:



Program "ASTEROID"

```
0010 CLR
0020 INPUT "VÄLJ TID (100-1000)", T1
0030 POKE (69,8)
0050 FOR I=1 TO 8
0060 READ U(I), V(I)
0070 NEXT I
0080 DATA 0,0,0,-1,0,0,-1,0,0,0,1,0,0,0,
     0,1,0,0
0090 CLR
0100 ?CHR¤(29):
0110 FOR I=1 TO 50
0120 X=RND*31+1
0130 Y=RND*31+1
0140 CUR(X,Y)
0150 ?"*";
0160 NEXT I
0170 GSTO(4)
0180 X=48: Y=64
0190 T0=T0+1
0210 IF T0=T1 THEN ?"FÄRDIG P=";
     N/50;"%";CHR¤(3):POKE(69,0):END
0220 K=KEY
0230 IF K=0 THEN 260
0240 S=S+U(K-48)
0250 T=T+V(K-48)
0260 X=X+S
0270 IF X>0 IF X<95 THEN 290
0280 S=-S: GOTO 190
0290 Y=Y+T
0300 IF Y>0 IF Y<127 THEN 320
0310 T=-T: GOTO 190
0320 GRCL(4)
0330 PLOT(X,Y)
0340 MOVE(X,Y)
0350 IF PEEK(66)=42 THEN 380
0360 MOVE(0,127)
0370 GOTO 190
0380 MOVE(X,Y):?" ";
0390 N=N+1
0400 MOVE(0,127):?N;
0410 GSTO(4)
0420 GOTO 190
```


Mycket snabb rörelse får man genom att rita delbilder och lagra med GSTO, varefter bilderna visas i snabb sekvens med GRCL (jämför tecknad film).

Exempel:

"ROTERANDE ROMERSKA RINGAR"

```
10 R=12
20 T=R*1.3
30 R1=1.5*R
40 R2=3*R
50 FOR I=0 TO 3
60 CLR
70 GOSUB 140
80 GSTO(I+3)
90 NEXT I
100 FOR I=3 TO 6
110 GRCL(I)
120 NEXT I
130 GOTO 100
140 F0=I*π/32
150 P=π/8
160 FOR F=0 TO 15
170 V=F0+P*F
180 K1=R*COS(V)
190 L1=T*SIN(V)
200 X=K1+R
210 X1=-K1+R+R1
220 X2=X+R2
230 X3=X1+R2
240 X4=X2+R2
250 Y=L1+45
260 Y1=L1+60
270 PLOT(X,Y1)
280 PLOT(X1,Y)
290 PLOT(X2,Y1)
300 PLOT(X3,Y)
310 PLOT(X4,Y1)
320 NEXT F
330 RETURN
```

Det tar ganska lång tid innan bilderna ritas upp!

Lösningen på detta är att till programmet lägga en tabell över sinusfunktionen i datasatser:

Exempel:

```
500 DATA 0, 0.195, 0.383, 0.556, 0.707
501 DATA .....
```

Cosinusfunktionen fås ur samma tabell!

Kapitel 9

Maskinspråksprogrammering

9.1 Varför maskinkod?

När man råkar ut för situationer som att lagringskapaciteten är för liten, en Basicrutin tar för lång tid eller kanske inte finns, kan man behöva gå över till att programmera i maskinspråk.

Mikroprocessorn i Esselte 100 är tillverkad av Motorola och heter MC6802.

För att underlätta programmering i maskinkod bör du ha tillgång till Motorolas programmeringsmanual för MC6800, samt någon lärobok om denna mikroprocessor, t ex *Lär dig använda mikrodatorn* (Andersson—Kullbjer 1979. DIDACT. Kan beställas från Esselte Studium). I appendix på sid 59—61 hittar du en sammanfattning av instruktionslistan för MC 6800.

I maskinkodsexemplen i det här kapitlet är adresser och data angivna i hexadecimal kod. I texten används suffixet H för att ange hexadecimal kod, t ex 39H.

9.1.1 Minneskarta

Adress (hex) Från	Till	Innehåll
0000	0036	RAM För användaren
0037	0140	RAM Monitor och Basictolk
0140	0286	RAM Buffertar till Basic
0287	1FFE	RAM Basicprogram + variabler
1FFF	Nedåt	RAM Stack
2000	BFFF	Oavkodat
C000	C3FF	RAM Bildskärm
C808	C80A	PIA Tangenbord, kassett etc
C810	C813	PIA Egen port, Realtidsklocka
D000	F120	ROM Basictolk
F121	FFFF	ROM Monitor, grafik etc

9.1.2 Maskinvara

Processor:	MC6802	
Klocka:	1MHz, cykeltid 1 μs	
RAM:	HM6116, MK4118	9 Kbyte
ROM:	TMS2716	14 Kbyte
PIA:	MC6821	

9.2 Hur man anropar maskinkodsprogram

9.2.1 Anrop från Basic-program

Vart och ett av följande kommandon ger uthopp från ett Basic-program till ett maskinspråksprogram:

A = USER (X)
A = INP (X)
A = TIME (X)
A = SYS (X,Y)
OUT (X,Y)
PATCH

9.2.2 A = USER (X)

Programmet fortsätter till en maskinkodsrutin, vars startadress finns lagrad i två minnesceller; 00FCH och 00FDH, dvs 252 och 253 decimalt (kallas USERPTR). I cellerna 00FEH och 00FFH, dvs 254 och 255 decimalt (kallas VARPTR), finns en pekare till ett flyttal i minnet. Detta är lika med talet X i USER (X). Maskinkodsrutinen skall avslutas med koden 39H, dvs 57 decimalt (detta är koden för instruktionen RTS. Körningen fortsätter då med nästa Basic-instruktion och det ovan nämnda flyttalet tilldelas variabeln A.

Exempel:

En maskinkodsrutin finns i minnet med start i 0800H. Denna rutin beräknar t ex 10-logaritmen av ett tal.

```
10 POKE (252,8):POKE (253,0)
20 A = USER (X)
```

Flyttalet lagras i BCD-form enligt:

T ex talet Pi = 3.14152965

Byte	0	1	2	3	4	5
	03	14	15	29	65	01

9.2.3 X = SYS (Y,Z)

Detta är ett systemkommando som vid uthopp tar med sig värdet av två variabler Y och Z och återvänder med variabeln X. Alla tal är heltal mellan noll och 255. Rutinen skall starta i adressen 0000H. Vid uthoppet kommer ACCA att få värdet av Y och ACCB värdet av Z, vid återhoppet kommer innehållet i ACCA att tilldelas X. SYS-rutinen skall avslutas med instruktionen RTS (39H).

Exempel:

En rutin som ger resultatet X = 1 om Y och Z är lika, 0 annars.

Maskinkodsrutin:

Adress	Kod
0000	11 CBA
0001	27 BEQ
0002	02
0003	4F CLRA
0004	39 RTS
0005	86 LDAA #
0006	01 1
0007	39 RTS

Basicprogrammet ser ut så här:

```
10 INPUT Y,Z
20 X = SYS (Y,Z)
30 ?X
```

9.2.4 A=INP (X)

Funktionen INP är till för att underlätta inhämtning av data från yttre enheter. Vid start är den inställd för att fungera tillsammans med parallellgränssnittet IO100, som finns som tillbehör, se 12.6. Användaren kan dock förändra funktionen genom att skriva en etta i bit 7 i adress 67 decimalt, t ex med POKE (67,128). INP-rutinen skall starta i adress 0009 och avslutas med RTS. Talet X är ett 16-bitars tal (0-65535). De 8 mest signifikanta bitarna överförs till ACCA och de 8 minst signifikanta till ACCB. Det tal som vid återhoppet förs till variabeln A är ett 8-bitars tal hämtat från ACCA.

Exempel:

Maskinkod

Adress		Kod		
000F	D7 21	STAB	TEMP +1	
0011	8B C0	ADDA #	BEG	
0013	97 20	STAA	TEMP	
0015	DE 20	LDX	TEMP	
0017	A6 00	LDAA	0,X	
0019	39	RTS		

I Basicprogrammet:

```
90 POKE (67,128)
100 A=INP (56)
```

Om inte någon egen funktion har definierats, kommer A=INP (X) att peka ut kanal X och därifrån hämta ett tal A från IO100.

9.2.5 OUT (X,Y)

Funktionen OUT har flera funktioner och måste därför föregås av ett annat kommando, IO, APULS, BPULS, eller en egen OUT-rutin. Talen X och Y är 8-bitars tal (0-255).

9.2.6 IO

Efter kommandot IO medför OUT (X,Y) att talet Y matas ut på kanal X på IO100.

9.2.7 APULS och BPULS

Kommandot APULS resp BPULS före OUT (X,Y) medför att X resp Y ms långa pulser kommer att matas ut på utgång A resp B. Genom att ställa bit 1 i adress 44H (=68 decimalt) kommer man att kunna variera tidbasen. Tidbasen, som normalt är 1 ms, kan ändras mellan 40 μs och 0,5 s. Detta sker genom att man skriver in ett 2-bytes tal i cellerna 004FH och 0050H, dvs 79 och 80 decimalt. Talet 0 i adress 79 och 123 i adress 80 ger tidsbasen 1 ms.

9.2.8 Egna OUT-rutiner

Genom att skriva önskad startadress i USERPTR (adresserna 252 och 253 decimalt, se 9.2.2) kan man definiera en egen OUT-rutin. Talet X går till ACCA, talet Y till ACCB.

9.2.9 TIME (X)

TIME-funktionerna mäter tid och frekvens och återvänder med ett flyttal i variabeln T. TIME(17) är användardefinierad och startar i adress 000CH. Ingen variabel förs med, dock kan man förstås låta enskilda minnesceller användas till detta. Då resultatet skall läggas i den buffert som utpekas av VARPTR i flyttalsform, är denna funktion dock något besvärlig att använda.

Möjligheter finns dock att utnyttja en befintlig rutin för omvandling från 16-bitarstal till flyttal (FIXA-FLYT) vilket medför viss förenkling.

Exempel:

En rutin som mäter en negativ puls på IN B med upplösning 12 μs.

Maskinkod

Adress	Kod		
000C	CE FFFF	LDX #	FFFF
000F	B6 C80A	LDAA	PIAB
0012	B6 C80B	LDAA	PIABCR
0015	2A FB	BPL	
0017	B6 C80A	LDAA	PIAB
001A	08	INX	
001B	B6 C80B	LDAB	PIABCR
001E	2A FA	BPL	
0020	DF 58	STX	SPARX
0022	7E F12D	JMP	FIXAFLYT

Basicprogram:

```
10 A=TIME(17)*12E-6
20 ?A
```

Faktorn 12*E−6 i rad 10 gör att resultatet presenteras uttryckt i sekunder.

9.3 Maskinkodsmonitorn

Med PATCH-kommandot startas i datorn ett monitorprogram. Detta program låter dig studera och ändra i minnet. Du kan skriva enkla maskinkodsprogram och provköra dessa. Datorn skriver MON på bildskärmen då monitorn är beredd att ta emot ett kommando. Du har fyra kommandon att välja mellan, M, J, R och B.

M står för Memory och du skall efter M skriva in en adress i hexadecimal form. Datorn svarar med det hexadecimala data som finns på denna adress. Trycker du ned mellanslagstangenten får du se nästa adress och data osv.

Vill du studera föregående adress, trycker du på ⟨−⟩. När du vill ändra innehållet i en minnescell, dvs när du vill skriva in ett program, skriver du helt enkelt två hexadecimala siffror per adress. Genom att trycka ned ⟨RETURN⟩ kan du på nytt välja kommando.

J står för Jump, När du skrivit J väntar sig datorn en adress dit du vill att den skall "hoppa", dvs den adress där en av dig skriven maskinkodsrutin börjar. Datorn börjar nu utföra instruktionerna och först då den funnit instruktionen RTS (39H) kommer den tillbaka till MON.

Om du i ditt program skrivit in instruktionen SWI (3FH) kommer processorn att hoppa tillbaka till MON. Detta kallas att sätta en brytpunkt.

R står för Register. När du skrivit ⟨R⟩ kommer datorn att skriva ut de register som finns i stacken. Om du har haft en brytpunkt i ditt program kommer du med detta kommando att kunna examinera

ackumulatorer och register. Ordningen hos registren på bildskärmen är:

SP CC B A IX PC

B står för Back. Detta kommando för dig tillbaka varifrån du kom, normalt tillbaka till Basictolken. Om du använt en brytpunkt går datorn tillbaka till den instruktion som följer efter brytpunkten.

9.3.1 Sammanfattning av monitorkommandona

M hhhh	Se i minnesposition hhhh. Vill du ändra, skriv bara in det nya värdet!
Mellanslag	Se nästa adress.
−	Se föregående adress.
⟨RETURN⟩	Tillbaka till monitorn.
J hhhh	Starta maskinspråksprogram som startar i hhhh. Detta måste avslutas med ett RTS=39H.
R	Skriv ut register SP, CC, B, A, IX, PC. Har endast betydelse i samband med avbrott, om du t ex har skrivit SWI=3FH i ditt program kommer du till monitorn och kan med R se registren. Genom att du vet var stacken är, kan du ändra lämpliga register.
B	Återvänder till det program varifrån den kom (Basic-tolken, eller program som avbrutits med SWI eller IRQ).

Om du skriver fel, kan du inte använda tangenten (←), som du är van vid. Skriv färdigt och gör om kommandot. Enda undantaget är J. Skriver du nämligen fel hoppadress kan otrevliga saker hända med ditt program! Tryck då istället på CTRL-C innan du skrivit felaktiga hoppadressen, så kommer du tillbaka till Basic-tolken. Där kan du åter skriva PATCH och J samt rätt hoppadress.

9.3.2 Var får du skriva maskinkod?

Esselte 100 innehåller RAM-minne i adresserna 0000H−1FFFH. Nästan allt detta minne nollställs av Basictolken när man skriver kommandot RUN. Ett femtiotal byte, adresserna 0000H−0035H, har dock reserverats för maskinkod och nollställs inte! Maskinkodsprogram skrivs alltså lämpligen här!

Stora maskinkodsprogram, som inte får plats i den reserverade minnesarean, kan läggas högt upp i RAM, mellan 1800H och 1EFF. Observera att Basic-programmet då måste startas med GOTO XX, där XX är Basic-programmets första radnummer. GOTO nollställer inte minnet.

Om du inte använder Basic-tolken alls, har du nästan hela minnet ledigt för maskinkod, från 0300H−1FFFH (samt den reserverade arean, givetvis).

9.3.3 Jämförelse mellan Basic och maskinkod

Exempel:

Fyll bildskärmen med bokstaven B.

I Basic:

```
10 V1=12*4096:V2=V1+1023
20 FOR U=V1 TO V2
30 POKE(U,66)
40 NEXT U
```

Programmet tar ett tiotal sekunder att köra!

I maskinspråk:

```
PATCH
M0000
0000   86 42      LDAA #'B      (Program)
0002   CE C000    LDX # C000
0005   A7 00      STAA 0,X
0007   08         INX
0008   8C C400    CPX # C400
000B   26 F8      BNE 0005
000D   3F         SWI           (Brytpunkt)
000E   39         RTS
J0000          (Starta programmet)
MON            (Brytpunkt=SWI)
R              (Se på register)
1FF3 C0 00 42 C400 000A
MON
M1FF3          (Titta efter i stacken)
1FF3 C0
1FF4 00
1FF5 42 41     (Ändra ACCA till 'A)
B              (Tillbaka från brytpunkt)
MON            (Färdig)
B              (Till Basic)
ESSELTE 100
```

Tid ungefär en tiondels sekund, dvs ca 100 ggr snabbare än motsvarande Basicprogram!

Om man ersätter den första instruktionen med NOP (dvs data 01 i adress 0000 och 0001) i programmet kan man kalla på det med A=SYS(A,B). Variabeln kommer att fylla skärmen. Ta bort brytpunkten i adress 000DH genom att skriva RTS (39H) istället.

9.4 Att spara maskinkodsprogram på kassett

Ett maskinkodsprogram, som skrivits med monitorn enligt ovan, kan tyvärr inte sparas på band. Detta gör dock inte så mycket, man kan nämligen skriva in maskinkodsprogram med hjälp av Basic-instruktionen POKE.

Ovanstående maskinkodsprogram med NOP i början och utan SWI kan skrivas så här:

```
10 POKE(0,1)
20 POKE(1,1)
30 POKE(2,206)
osv.
```

Totalt fordras 15 POKE-instruktioner.

Enklare är att använda en loop och DATA-satser. Hela programmet ser då ut så här:

```
10 FOR I=0 TO 13
20 READ A
30 POKE(I,A)
40 NEXT I
50 DATA 01, 01, 206, 192, 0, 167, 0, 8
60 DATA 140, 196, 0, 38, 248, 57
```

Körs detta med RUN, matas maskinkoden in på rätt plats i RAM. Maskinkodsprogrammet körs sedan med kommandot SYS.

9.5 Subrutiner i Basic-tolken

9.5.1 Utmatningsrutiner

Adress	Namn	Funktion
F806	TTY1UT	Skriver ACCA på bildskärmen, ASCII-kod
F80C	TTY2UT	Skriver ACCA till serieporten, testar Ain
F812	TTY3UT	Skriver ACCA till Band, Kansas City
F818	TTY4UT	Egen utmatning av ACCA, Start 0003
F81E	TTY5UT	Skriver med stora bokstäver, storlek i 0069
F82A	TTY7UT	Skriver serieut utan test av Ain
F863	CURMOVE	Ställer markören i rad B, kol A

9.5.2 Inhämtning av data

Adress	Namn	Funktion
F803	TTY01N	Hämtar ett tecken från tangentbordet till A. Sedan kommer ett automatiskt CR
F809	TTY1IN	Hämtar ett tecken från tangentbordet, editeringsfunktionerna fungerar
F80F	TTY2IN	Hämtar ett tecken från serieinporten = Bin
F81B	TTY4IN	Egen inhämtningsrutin skall starta i 0006
F827	TTY6IN	Hämtar ett tecken till ACCA från promkassett
F82D	TTY7IN	Hämtar ett tecken från serieingången Bin, talar om att den lyssnar med en etta på Aut
F830	KBDTST	Ger carry = 1 om en kontrolltangent tryckts ner samt motsvarande kod i ACCA
F866	INPUT	Skriver ut ett kanalnummer från ACCB och hämtar ett tal till ACCA från Ex parallellinterface
F86C	KEY	Om någon tangent är nedtryckt kommer ACCA att få dess ASCII-kod annars blir den noll. Obs, mellanslag, shift och kontroll går ej!

9.5.3 Grafiska rutiner

Bildskärmen tänks ha 96 * 128 rutor adresserade med ett X, Y par, 0 < = X < = 95, 0 < Y < = 127.

Vid uthopp till grafiska rutiner antages ACCA innehålla X-koordinaten och ACCB Y-koordinaten.

Adress	Namn	Funktion
F836	AXIS	Ritar axlar på bästa möjliga position
F839	BAR	Ritar staplar
F83C	DRAW	Drar en linje från sist satta punkt
F83F	MOVE	Flyttar markören
F842	PLOT	Sätter om möjligt en punkt
F845	GSTO	Lagrar bildskärmens innehåll i en av 6 minnen
F848	JSEC	Ritar sektordiagram % ACCB
F869	JRCL	Hämtar en ny sida till bildskärmen
F86F	JCLR	Suddar alla tecken på bildskärmen vars ASCII-kod ligger mellan ACCA och ACCB.

9.5.4 Extra funktioner

Adress	Namn	Funktion
F833	PATCH	Hopp till Monitor
F851	MASTER	Hopp till kommunikationsrutin
F85A	TERM	Hopp till Terminalemuleringsrutin
F85D	TIME	Hopp till tidmätningsrutin, ACCB visar vilken 0—17
F857	OUT	Utmatningsprogram
F860	WAIT	Väntar tiden 256 * ACCA + ACCB ms
F872	RESET	Kall reset, allt nollställs
D000	BASIC	Kall start Basic programarea nollställs
D003	BASIC1	Varm start av Basic.

9.5.5 Beräkningsrutiner

Följande rutiner utför vanligt förekommande uppgifter. Används som subrutiner i egna program.

Adress	Namn	Funktion
F8EA	CA2HI	Sätter Aut hög
F8CB	WRITAP	Skriver ACCA till band, kansas-city
F922	CA2LO	Sätter Aut lås
F946	TAPIN	Läser ett tecken från band till ACCA
F9C3	PRIUT	Skriver ACCA till serieutgången, But. Ingen test av Ain eller fördröjning
FB9C	HEXBCD	Omvandlar ett 2 byte binärt tal till ett flyttal i samband med t ex USER, binärtalet skall ligga i 0058,0059
FBE6	PULSA	Sänder ut ACCA st ACCB ms långa pulser på Aut
FC0E	PULSB	Sänder ut ACCA st ACCB ms långa pulser på But
FC68	SERIN	Läser in ett seriellt tecken utan att invänta någon startbit
F132	PDATA	Skriv en sträng so pekas ut av IX t o m ett (04)
F13F	P2HEX	Skriv ut ACCA som två ASCII-tecken
F149	HEXASC	Omvandla från ASCII till HEX i ACCA
F153	ASCHEX	Omvandla från Hex till ASCII i ACCA
F15E	IN2HEX	Hämtar in en byte till ACCA
F1CF	BYGGAD	Hämta i en hexadress till IX
F1DC	P4P2	Skriv adress och data som IX pekar på
F1F2	NYRAD	Ge ny rad
F1FD	PSPACE	Skriv mellanrum
F283	DECIND	Minskar indexregistret med 32
F2BD	GFRAFOK	Är A,B inom grafikintervallet
F7CD	IN81	Läser IX st data från serieinterface
F7E0	UT81	Skickar ut ACCA i serieform, synkront
F7F3	PULS	Ger en 8 μs lång puls på Aut

Kapitel 10

Databaser

10.1 Databashantering med Esselte 100

Att samla data i s k databaser och bearbeta dessa data är en mycket vanlig uppgift för datorer. Normalt kräver detta massminnen med mycket hög kapacitet, eftersom en databas kan innehålla miljontals tecken.

Till Esselte 100 används normalt en vanlig kassettbandspelare som massminne. Kapaciteten hos ett kassettband av typ C30 är ca 50 000 tecken, vilket motsvarar ungefär 10—20 fulla A4-sidor.

Att lagra program med Esselte 100 är enkelt, kommandot SAVE lagrar hela det aktuella programmet. Att lagra data är däremot lite besvärligare, och en speciell teknik, som kommer att beskrivas i de följande avsnitten, måste användas.

10.2 Att lägga upp register

Operativsystemet i Esselte 100 innehåller inte någon filhantering, men man kan ändå relativt enkelt bygga upp mindre register och databaser och lagra dessa på band. Detta sker genom användning av DATA-satser. Ett enkelt register över t ex IC-kretsar, kan se ut så här:

```
1000 DATA SN7400,Texas,10
1001 DATA SN7402,Fai.,7
1002 DATA SN7405, Texas,2
1003 DATA SN7447,Mot.,22
```

Radnumret R får fungera som datapekare, en inläsning av en post kan då göras så här:

```
100  RESTORE R
110  READ A¤,F¤,N
```

Först pekas rad R ut, varefter namn, fabrikat och antal läses in i respektive A¤,F¤ och N.

För att datorn ska slippa läsa igenom alla poster för att hitta en post, kan i huvudsak två olika metoder användas:

1. Låt de första siffrorna i DATA-satsens nummer vara lika med ASCII-koden för första bokstaven i postens namn.
2. Bygg upp ett separat nyckelregister.

Själva dataregistret kan se ut hur som helst, strängar och tal kan blandas.

Exempel:

```
1008 DATA Lars-Erik Björklund
     Rydsvägen 356B, 58250 LINKÖPING
1009 DATA Sven Karlsson Grenstigen 12
     823 01 Nystad
```

Varför är det egentligen så bra att använda DATA-satser?

1. En datafil kommer att se ut som ett program och detta gör att det kan lagras på band som ett program, med kommandot SAVE. Det kan hämtas tillbaka med LOAD eller APPEND.
2. DATA-satser medger mycket effektiv lagring av strängar i Esselte 100. Det beror på att strängar måste dimensioneras och det är den största som bestämmer över de övriga. DATA-satser kan vara olika långa.
3. Radnumret ger lätt access till önskad post och medger lagring av upp emot 9 000 poster (om minnet räcker!)
4. Basic-editorn kan användas vid inskrivning av nya poster. Alla kommandon kan användas: LIST, DEL, REN etc.
5. Ett litet enkelt maskinkodsprogram kan tillverka DATA-satser och tillfoga dessa till programminnet, så att t ex mätvärden från ett försök automatiskt överförs till DATA-satser.

Då databasen ska lagras, minskas lämpligen fördröjningen till kassettutmatningen från 12 till 4 null-tecken.

Data lagras alltså i form av DATA-satser i ett program. Med hjälp av ett litet program kan datorn själv fås att lägga till, ta bort eller ändra i DATA-satserna.

Då en DATA-sats ska skapas, lagras först i RAM-minnet en exakt kopia av raden.

Exempel:

Data bestående av ett personnamn och ett telefonnummer ska lagras i en DATA-sats med radnummer 2000. I RAM-minnet lagras då texten

```
2000 DATA Björklund 173790
RUN
```

När detta gjorts, ska den aktuella inläsningsporten (port 1) bytas mot en port som läser direkt ur RAM-minnet (port 4), varefter programmet stoppas. Datorn läser nu ur RAM in den önskade DATA-raden, som placeras in på rätt ställe i programmet. Sedan läser datorn kommandot RUN och startar körning av programmet. På programmets första rad måste kommandot PORT = 1 stå.

Så här ser det hela ut:

```
  5  PORT=1
 10  GOSUB 100
 20  LINPUT"NAMN NR",A¤
 30  RESTORE 180
 40  READ N
 50  ?#4,N;"DATA";A¤
 60  ?#4,"1000 DATA";N+1
 70  ?#4,"RUN"
 80  PORT=4
 90  END
180  DATA 1000
```

För att port 4 ska få den önskade funktionen måste dessutom ett litet (23 byte) maskinkodsprogram laddas in. Detta sker lämpligen med DATA-satser och POKE-satser enligt följande:

```
100  RESTORE 160
110  FOR I=3 TO 25
120  READ A
130  POKE(I,A)
140  NEXT I
150  RETURN
160  DATA 32,9,0,222,22,166,0,9,223,22,57,
     222,24
170  DATA 167,0,9,223,24,57,29,0,29,0
```

Den här metoden för generering av DATA-satser har ett mycket vidsträckt användningsområde, t ex vid mätning och datainsamling eller vid uppläggning av mera "normala" databaser för lagerbokföring etc.

10.3 Mätvärdesinsamling

Ett program som hämtar in mätvärden och lägger upp dem i DATA-satser ser ut så här:

```
 10 FOR I=3 TO 25
 20 READ A
 30 POKE(I,A)
 40 DATA 32,9,0,222,22,166,0,9,223,22,57,
    222,24
 50 DATA 167,0,9,223,24,57,29,0,29,0
 60 PORT=1
 70 T=TIME(3)
 80 RESTORE 1000
 90 READ N
100 ?#4,N;"DATA";N;T
110 ?#4,"1000 DATA";N+1
120 ?#4,"RUN"
130 PORT=1
140 END
1000 DATA 2000
```

Raderna 10 t o m 50 definierar port 4 enligt förra avsnittet, mätningen görs på rad 70 (givetvis kan vilken mätinstruktion som helst användas här) och resten av programmet placerar insamlade data i DATA-satser, med början i rad 2000. Finessen är att mätdata efter avslutad mätning lätt kan lagras på band, med kommandot SAVE. Även programmet lagras, vill man bara ha med DATA-satserna får man först radera programmet med DEL 1,1000 före lagringen. Data kan behandlas vid ett senare tillfälle, genom att de då hämtas från bandet med kommandot LOAD eller APPEND, beroende på om databehandlingsprogrammet matas in efter eller före data.

10.4 BAS100, ett databashanteringsprogram

BAS100 är ett databashanteringsprogram för Esselte 100. Data kan läggas till, ändras och sökas i en databas.

BAS100 består av två delar, en programdel och en datadel. Båda dessa ingår dock i ett och samma Basic-program. Lagring sker på en vanlig kassett (dock bör program och data lagras på olika kassetter!)

Bruksanvisning:

1. Ladda in programdelen via band, tangentbord eller PROM.
2. Lägg till datadelen med kommandot APPEND (behövs givetvis inte om basen är tom).
3. Starta programmet med RUN.

På bildskärmen skrivs nu en meny ut, ur vilken önskat kommando kan väljas. Hela registret kan skrivas ut, det går att läsa framåt eller bakåt, man kan lägga till och ändra poster, söka efter poster med hjälp av nyckelord, samt lagra data på band.

Med kommandot P skrivs hela basen ut på skärmen, med kommandona N eller U kan post för post studeras (Ned respektive Upp). Kommandot Å används vid ändring och S vid sökning. Kommandot T används vid lagring på band. Kommandot L medför att ett frågetecken skrivs ut och datorn frågar efter en ny post. Två varianter finns:

I. Skriv i en följd in posten utan kommatecken.
II. Skriv in delposter med komma mellan.

Exempel:

I. Björklund Lars Rydsvägen 356B
II. Björklund, Lars, Rydsvägen 356B

Vid utskrift kommer mod II att tabuleras i kolumner, 9 tecken breda.

Kommandot S ger möjlighet att söka i databasen. Som svar på frågan Sökord ger man det ord/namn man söker. Om databasen består av poster utan komma (I), kommer programmet att söka genom alla posters alla tecken. Då det sökta ordet påträffas första gången, kommer denna post att skrivas ut, och sökningen fortsätter till slut. En databas av typ II medger snabbare sökning, då man förutom sökordet också anger i vilken kolumn det ska sökas. Kolumnen anges genom att specificera antalet komma före ordet.

Exempel:

I:Sökord? KALLE
ANDERSSON KALLE 172790
JANSSON KALLE 132456
SLUT

II.Sökord?,,13
JANSSON, KALLE, 132456
ERIKSSON, JAN, 141322
SLUT

Observera att sökning enligt fall II enbart sker i angiven kolumn.

Då någon post ska ändras, t ex i en lagerdatabas, används först sökkommandot S för att hitta posten (eventuellt N eller U). Sedan ges kommandot Å. Den angivna radenm skrivs ut på skärmen och markören placeras automatiskt i början på raden. Med ⟨ – ⟩ kan nu raden kopieras och ändras efter behov. Då hela raden är färdig, trycker man på ⟨RETURN⟩.

Då man vill spara sin databas på band finns två varianter:

A: Om bara databasen ska sparas, ges kommandot T och bandspelaren startas. Programmet rensas nu från programrader och bara dataraderna sparas på bandet.

B: Stoppa programmet med CTRL-C och spara hela programmet med SAVE.

Inga viktiga variabler förstörs vid stoppet, utan programmet går att köra med RUN igen.

Hur lagras data?

Exempel:

Post:

ANDERSSON,ANDERS,UGGLEBOV,123456

lagras som

2007 DATA ANDERSSON,ANDERS,UGGLEBOV, 123456, #

På rad 1000 pekas nästa lediga plats ut:

1000 DATA 2008

Några typiska användningsområden för BAS100 är t ex adressregister, lagerregister, litteraturregister etc.

BAS100 innehåller ingen sorteringsfunktion, men ett separat program kan sortera datafilen. Inga försök till uppsnabbning av sökrutinen har gjorts, utan programmet ska ses som ett undervisningshjälpmedel.

Program BAS100

```
0005 !BAS100
0010 PORT=1
0015 GOSUB 435
0020 POKE(78,6)
0025 POKE(9*16,64)
0030 LINE=0:T9=9
0035 P0=2000:!POINTER
0040 GOTO 275
0045 !LÄGG TILL POST
0050 IF FRE<1000 THEN ?"FULLT
     MINNE":GOTO275
0055 LINPUT A¤
0060 ?#4,N;"DATA";A¤;",#"
0065 ?#4,"1000DATA";N+1
0070 ?#4,"RUN"
0075 PORT=4
0080 END
0085 !ÄNDRA
0090 IF P0>1999 IF P0<N THEN100
0095 GOTO 275
0100 RESTORE P0
0105 ?:?"  ";
0110 READ A¤
0115 IF A¤="#"THEN130
0120 ?A¤;",";
0125 GOTO 110
0130 ?CHR¤(8);CHR¤(14);CHR¤(13);
0135 LINPUT B¤
0140 ?#4,P0;"DATA";B¤;",#"
0145 ?#4,"RUN"
0150 PORT=4
0155 END
0160 !SÖK
0165 K1=1
0170 LINPUT "SÖKORD",X¤
0175 IF LEFT¤(X¤,1)="","THEN K1=
     K1+1:X¤=MID¤(X¤,2,64):GOTO175
0180 X=LEN(X¤)
0185 L0=2000
0190 FOR L=L0 TO N-1
0195 RESTORE L
0200 FOR K=1 TO K1
0205 READ A¤
0210 IF A¤="#" THEN K=K1+1
0215 NEXT K
0220 IF K>K1+1 THEN 260
0225 A=LEN(A¤)
0230 IF A<X THEN260
0235 S=A-X+1
0240 FOR T=1 TO S
0245 IF MID¤(A¤,T,X)=X¤
     THEN T=S+1
0250 NEXT T
0255 IF T>S THEN P0=L:RESTORE
     L:?:GOSUB360
0260 NEXT L
0265 ?:?"SLUT"
0270 GOTO 295
0275 !MENY
0280 CLR
0285 ?"BAS 100"
0290 ?"----------"
0295 ?:?"LÄGG TILL,ÄNDRA,SÖK,
     PRINT,TAPE,NED,UPP"
0300 RESTORE 1000
0305 READ N
0310 INPUT#0,X¤
0315 IF X¤="" "THEN280
```

```
0320 IF X¤="N"THEN510
0325 IF X¤="U"THEN505
0330 IF X¤="T"THEN475
0335 IF X¤="L" THEN45
0340 IF X¤="P" THEN405
0345 IF X¤="Ä"THEN85
0350 IF X¤="S"THEN160
0355 GOTO 285
0360 !SKRIV POST
0365 P9=0
0370 READ A¤
0375 IF A¤="#"THEN 395
0380 ?TAB(P9*T9);A¤;
0385 P9=P9+1
0390 GOTO 370
0395 ?
0400 RETURN
0405 CLR:!SKRIV UT
0410 FOR L=2000 TO N-1
0415 RESTORE L
0420 GOSUB 360
0425 NEXT L
0430 GOTO 295
0435 DATA 32,9,0,222,22,166,0,9,223,22,57,
     222,24
0440 DATA 167,0,9,223,24,57,29,0,29,0
0445 RESTORE 435
0450 FOR I=3 TO 25
0455 READ A
0460 POKE(I,A)
0465 NEXT I
0470 RETURN
0475 INPUT "SLÅ TILL BANDSPELAREN
     〈RETURN〉",A¤
0480 ?#4,"DEL10,999"
0485 ?#4,"SAVE"
0490 ?#4,"PORT=1"
0495 PORT=4
0500 END
0505 P0=P0-1:GOTO 515
0510 P0=P0+1:!NED
0515 IF P0>N-1 THEN P0=N-1
0520 IF P0<2000 THEN P0=2000
0525 RESTORE P0
0530 ?CHR¤(8);
0535 GOSUB 360
0540 GOTO 310
1000 DATA 2000
```


Kapitel 11

Digitala in- och utsignaler

11.1 Digitala signaler via PRINTER-kontakten

I PRINTER-kontakten finns 4 datakanaler tillgängliga, IN A, IN B, UT A och UT B. Utkanalerna kan noll- eller ettställas och inkanalerna avläsas.

Exempel: Sätt UT A=1 och UT B=0

```
10 POKE(51209,60)   ger etta ut på UT A
20 POKE(51211,52)   ger nolla ut på UT B
```

Exempel: Hämta in data till variabeln X

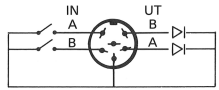
```
10 Y=PEEK(51220)        läser in data till Y
20 X=AND(Y,16)/16       X=0 om IN A=0,
                        X=1 annars
30 Z=AND(Y,32)/32       Z=0 om IN B=0,
                        Z=1 annars
```

Alternativt kan rad 20 och 30 ersättas av

```
20 U=AND(Y,48)/16
```

IN A=0, IN B=0 ger ger U=0
IN A=1, IN B=0 ger U=1
IN A=0, IN B=1 ger U=2
IN A=1, IN B=1 ger U=3

Prova gärna med följande koppling!



Låt oss dröja kvar ett tag vid kopplingen. Ett program, som känner av om strömbrytaren A är sluten ("0") och i så fall får lysdioden B att blinka 3 gånger med frekvensen 2 Hz, kan se ut så här:

```
10  Y=PEEK(51210)
20  X=AND(Y,16)/16
30  IF X=1 THEN 10
40  FOR I=1 TO 3
50  POKE(51211,60)
60  WAIT(250)
70  POKE(51211,52)
80  WAIT(250)
90  NEXT I
100 GOTO 10
```

Att generera ett pulståg på det här sättet fordrar tydligen ganska många instruktioner. Esselte 100 har därför en speciell instruktion, OUT(X,Y), som genererar X fyrkantpulser med pulslängd Y ms. OUT(X,Y) måste föregås av kommandot APULS eller BPULS, för val av utgång. Antalet pulser är begränsade till 255 och tiden till 255 ms. OUT-instruktionen förenklar programmet ganska avsevärt:

```
10 Y=PEEK(51210)
20 IF AND(Y,16)=16 THEN 10
30 BPULS
40 OUT(3,250)
50 GOTO 10
```

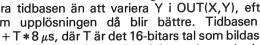
Ibland vill man ha andra pulstider, än de som normalt kan genereras. Tidbasen kan då ändras på följande sätt:

```
POKE(68,2)
POKE(79,A)
POKE(80,B)
```

Normalt är A=0 och B=124, vilket ger tidbasen 1 ms. Andra värden kan väljas enligt tabellen:

Tidbas ms	A	B
0,05	0	1
0,1	0	7
1	0	124
10	4	216
100	48	112
500	242	48

Har man t ex valt A=48 och B=112, dvs tidbasen 100 ms, ger OUT(5,25) 5 st pulser, var och en med längden 2,5 sekunder.



I vissa fall (t ex vid tonalstring) är det bättre att variera tidbasen än att variera Y i OUT(X,Y), eftersom upplösningen då blir bättre. Tidbasen=-40+T*8 µs, där T är det 16-bitars tal som bildas av de två talen A och B enligt tidigare. Inom det hörbara området är A=0, dvs T=B.

11.2 Extra parallellport

På huvudkortet till Esselte 100 finns en 40-polig sockel (U401), se avsnitt 1.4, där användaren kan placera ytterligare en parallellport (PIA) MC6821. Den innehåller 2 st 8-bitars portar samt 4 st kontrollsignaler. Portarna är programmerbara för att kunna användas som in- eller utgångar.

Vid tillslag programmeras automatiskt denna krets så att port A blir ingång och port B utgång. De två flankkänsliga ingångarna CA1 och CB1 kommer att vara känsliga för negativ flank. De två återstående kontrollsignalerna CA2 och CB2 blir utgångar som läggs låga (0 V).

Adresser i PIA:n

ORA, DDRA	51216
CRA	51217
ORB, DDRB	51218
CRB	51219

Användaren kan direkt läsa av A-porten med PEEK (51216) och skriva i B-porten med POKE (51218,X).

Det står användaren fritt att programmera om PIA:n efter eget behov. För detta fordras tillgång till databladet för MC6821.

Portarna finns tillgängliga på en 24-polig sockel (J401), alldeles bredvid PIA:n. Yttre enheter kopplas in via en 24-polig DIP-kontakt och en flatkabel.

Sockeln har följande pinkonfiguration:

Signal	Pin
PA0	1
PA1	2
PA2	3
PA3	4
PA4	5
PA5	6
PA6	7
PA7	8
PB0	9
PB1	10
jord	11
jord	12
PB2	13
PB3	14
PB4	15
PB5	16
PB6	17
PB7	18
CB1	19
CB2	20
CA1*	21
CA2	22
+5 V	23
+5 V	24

*OBS! CA1 kopplad till 50 Hz från videodelen.

11.3 Realtidsklockan

Till den extra PIA:ns ingång CA1 finns anslutet en 50 Hz klocksignal från videodelen. Klockan är kristallstyrd. Genom att ställa om PIA:n så att den kan generera avbrott, kan man starta en realtidsklocka. När som helst under programmering eller programkörning kan man få aktuell tid genom att skriva PRINT TIME(9).

Klockan ger tiden som ett sexsiffrigt tal, t ex 210735, angivet på för 24-timmarsklockor konventionellt sätt i timmar, minuter och sekunder.

Tiden kan, om så önskas, sparas i en variabel, t ex T=TIME(9). Det går också lätt att göra realtidsstyrsystem med hjälp av realtidsklockan, man kan då använda instruktioner av typen IF TIME(9) > 120020 THEN . . .

Innan klockan används måste den ställas. Detta kan göras antingen i maskinkod eller med ett litet Basic-program. Vi tittar först på hur klockan kan ställas i maskinkod.

Använd kommandot PATCH och skriv direkt i minnet:

PATCH
M 003C
15 om timman är 15
23 om minuten är 23
40 om klockan startas vid sekund 40
32 konstant
⟨RETURN⟩
M C811
35 sista "femman" skrivs 1 sekund före avsedd tid
⟨ ⟩
B tillbaka till Basic

Så här kan klockan ställas med en Basic-subrutin:

```
10 !STÄLL KLOCKAN
20 INPUT "TIM,MIN,SEK",T,M,S
30 T1=INT(T/10)
40 T=T1*6+T
50 M1=INT(M/10)
60 M=M1*6+M
70 S1=INT(S/10)
80 S=S1*6+S
90 POKE(60,T):POKE(61,M):POKE(62,S):
   POKE(63,50)
100 POKE(51217,53)
110 RETURN
```

Om du kontinuerligt vill visa klockan i bildskärmens övre vänstra hörn, med medelstora siffror, kan du lägga till

```
 5 GOTO 200
200 GOSUB 10:! STÄLL
210 SIZE=1
220 ?CHR¤(29);
230 CUR(1,5)
240 ?#5,TIME(9);
250 GOTO 230
```

Tänk på att realtidsklockan bara går att använda om den extra PIA:n är montrad!

Eftersom klockan styrs av en 50 Hz-signal, går det att få bättre tidsupplösning än 1 sekund. I minnescellen med adress 63 räknas 50 Hz-signalen ned till 1 sekund, från 50 till 0. Genom att läsa i denna minnescell med PEEK(63), kan man få tidsupplösningen 0,02 sekund.

11.4 Realtidsklockan som stoppur

Realtidsklockan kan fås att visa sekunder (0—239999) genom POKE(59,0). POKE(59,96) återställer till timmar, minuter och sekunder igen.

Nollställning av stoppuret:

POKE(63,50)
POKE(60,0)
POKE(61,0)
POKE(62.0)

Start av stoppuret:

POKE(51217,53)

Stopp av stoppuret:

POKE(51217,52)

Avläsning av stoppuret:

T=TIME(9)

Avläsning av stoppuret med 0,1 sekunders upplösning:

```
DIGITS=1
S=PEEK(63)
T=TIME(9)
T=T+(1-0.02*S)
```

Kapitel 12

Mätning och styrning via printerkontakten

12.1 Serie- och parallellgränssnitt

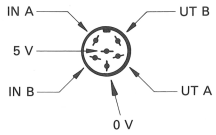
Överföring av information mellan datorn och yttre enheter kan ske antingen seriellt, som pulståg, eller parallellt, som ett 8-bitars binärt tal. Som standard är datorn försedd med ett seriegränssnitt, åtkomligt via PRINTER-kontakten. Totalt finns två ingångar och två utgångar tillgängliga här, och genom att Basic-tolken innehåller kraftfulla I/O-instruktioner, utgör PRINTER-kontakten ett gott alternativ i många mätsituationer.

Vid krav på snabbare I/O-hantering fordras ett parallellgränssnitt. Datorn är förberedd för detta (en krets och en flatkabel ska pluggas in på kretskortet), se vidare kapitel 11.

I fortsättningen handlar det här kapitlet om hur PRINTER-kontakten ska utnyttjas vid mätningar.

12.2 PRINTER-kontakten

PRINTER-kontakten är försedd med två ingångar och två utgångar, samt matningsspänning (5 V) och jord.



För den tekniskt intresserade kan nämnas att samtliga datakanaler går till en PIA (U402 i kopplingsschemat i avsnitt 1.4), IN A till PB4 och CA1, IN B till PB5 och CB1, UT A till CA2 och UT B till CB2. Om så önskas kan man med en liten maskinspråksrutin eller några POKE-satser programmera om PIA:n till godtycklig kombination av in- och utgångar.

Både in- och utgångar är TTL-kompatibla, dvs en insignal mellan 0 och 0,8 V tyds som en "nolla" och en insignal mellan 2,4 V och 5 V som en "etta". Maximal tillåten insignal är 5 V. Utgångarna kan sänka maximalt 3,2 mA.

12.3 Tidmätning

12.3.1 Tidmätningsinstruktioner

I Basic-tolken finns tio instruktioner för tidmätning. Dessa instruktioner tilldelar en variabel resultatet av en viss typ av tidmätning. Tiden erhålls i sekunder, med upplösningen 0,1 ms. Maximal mättid är 499999,9999 sekunder och precisionen är 0,1%. Då mätning pågår tänds klocksymbolen längst uppe till höger på bildskärmen.

Instruktionerna är av typen T = TIME(X), där T är variabeln och X är ett heltal, som anger typen av mätning. Följande instruktioner finns:

TIME(0) Mäter tiden från det instruktionen påbörjas till dess ingång A aktiveras av negativt gående flank (5 V till 0 V).

TIME(1) Mäter tiden som förflyter från det ingång A aktiverats till dess ingång B aktiverats av negativt gående flanker (5 V till 0 V).

TIME(2) Mäter tiden som förflyter mellan två aktiveringar av ingång A (negativt gående flanker, 5 V till 0 V), dvs periodtiden.

TIME(3) Mäter tiden mellan negativt gående flank (5 V till 0 V) och positivt gående flank (0 V till 5 V) på ingång A.

TIME(4) Mäter tiden mellan positivt gående flank (0 V till 5 V) och negativt gående flank (5 V till 0 V) på ingång A.

TIME(10) Samma som TIME(0), men ingång B.

TIME(11) Samma som TIME(1), men A och B växlade, dvs tiden mäts från att ingång B aktiverats tills ingång A aktiverats.

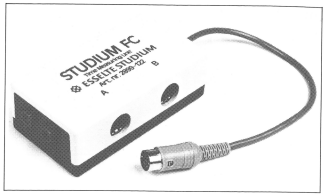
TIME(12) Samma som TIME(2), men ingång B.

TIME(13) Samma som TIME(3), men ingång B.

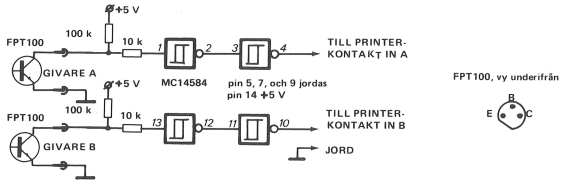
TIME(14) Samma som TIME(4), men ingång B.

12.3.2 Gränssnitt för tidmätning

I många fall fordras ingen som helst mätvärdesomvandlare vid tidmätning, t ex då givarna har TTL-utgång, kan sänka 1,6 mA och har stig- och falltider som är mindre än 1 μs. Ibland kan ett enkelt gränssnitt erfordras, exempelvis då fototransistorer används som givare. Ett sådant gränssnitt, Studium FC, avsett för mikrofoner och fototransistorer, kan köpas som tillbehör till Esselte 100. Studium FC är avsedd att användas vid fysikförsök i skolan, t ex för hastighetsmätningar.



Man kan mycket lätt själv bygga ett gränssnitt, avsett för fototransistorer, se kopplingsschema:



Endast en billig CMOS-krets (MC14584), fyra motstånd och en 6-polig DIN-kontakt behövs. Som givare används fototransistorn FPT100, avskärmad för ströljus genom att den placeras i botten av ett några centimeter långt rör (∅6 mm innerdiameter).

12.3.3 Programexempel

Ett program, som mäter tiden för en passage från givare A till givare B blir ovanligt enkelt!

```
10 ?TIME(1)
```

Vill man lägga upp en tabell för t ex 10 mätningar, kan programmet se ut så här:

```
10 FOR I=1 TO 10
20 T(I)=TIME(1)
30 NEXT I
```

Tabellen kan sedan användas för beräkning av medelvärde och spridning, diagramritning osv.

Vill man mäta hastighet går detta också bra, förutsatt att avståndet mellan givarna är känt. Exempelvis avståndet 2,5 dm ger

```
10 ?0.25/TIME(1)
```

Här erhålls hastigheten i meter per sekund.

Även acceleration kan mätas, om dessutom mätobjektets längd är känd. Antag att längden är 5 cm och att givaravståndet är 25 cm. Om likformigt accelererad rörelse förutsätts, kan sambandet

$s = v_0 t + a\,t^2/2$

utnyttjas. Löses accelerationen a ut ur detta uttryck erhålls

$a = 2(s - v_0 t)/t^2$

Program:

```
10 T1=TIME(3)
20 T2=TIME(10)
30 V0=5/T1
40 A=2*(25-V0*T2)/(T2*T2)
50 ?A
```

På rad 10 mäts den tid T1 mätobjektet blockerar fotocell A, på rad 20 mäts den tid T2 som förflyter från det att A lämnats till dess fotocell B påträffas. På rad 30 beräknas utgångshstigheten V0.

Vid skolförsök kan man med fördel utnyttja möjligheten att skriva med stora tecken på bildskärmen. Följande programrad mäter den tid en fototransistor, ansluten till ingång A, är belyst. Resultatet presenteras med siffror som är 4× normal storlek.

```
10 SIZE=1:?#5,TIME(3)
```

SIZE=2 ger 8× normal storlek, men då måste antalet siffror begränsas med kommandot DIGITS=3.

12.4 Frekvensmätning

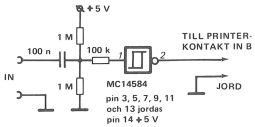
12.4.1 Frekvensmätningsinstruktioner

Basic-tolken i Esselte 100 innehåller två instruktioner för frekvensmätning, TIME(5) och TIME(8). Båda utnyttjar ingång B på PRINTER-kontakten och fordrar pulser med TTL-nivåer, maximal stig- och falltid $< 1\ \mu$s. I båda fallen erhålls frekvensen i Hz. Högsta mätbara frekvens är 32 kHz. TIME(5) räknar antalet pulser under 1 sekund och upplösningen är 1 Hz. TIME(8) räknar under 0,1 sekund, upplösning 10 Hz.

Vid mätning av mycket låga frekvenser är det bättre att mäta periodtid, se 12.4.3.

12.4.2 Gränssnitt för frekvensmätning

Kraven på stig- och falltider gör att det inte går att mäta t ex sinusformade signalers frekvens utan att signalen först pulsformas. En CMOS-krets går bra att använda (MC14584):



Insignalen kan här ha godtycklig kurvform. Amplituden måste vara större än ca 2 V. Vill man mäta frekvens på lågnivåsignaler, måste signalen förstås förstärkas före pulsformningen, t ex med en operationsförstärkare eller ett enkelt transistorsteg.

Om 100 nF-kondensatorn byts ut mot 4700 pF/3 kV, är för övrigt kretsen i kopplingsschemat användbar även vid mätning med GM-rör. GM-röret kopplas till högspänningen (några 100 V) via ett motstånd på 4,7 MΩ, och signalen tas ut över GM-röret.

12.4.3 Programexempel

Även program för frekvensmätning (ingång B) blir synnerligen enkla på Esselte 100:

```
10 ?TIME(5)
```

Mätningen tar 1 sekund och ger 1 Hz upplösning. Har man större krav på snabbhet, kan man istället skriva

```
10 ?TIME(8)
```

som bara tar 0,1 sekund, men i gengäld bara ger upplösningen 10 Hz.

Vill man mäta frekvensen på mycket lågfrekventa förlopp, är den här metoden inte så lämplig. Det hinner ju inte komma så många pulser under den förhållandevis korta mättiden, varför relativa felet blir onödigt stort. Det är då bättre att mäta periodtiden istället, och låta datorn beräkna frekvensen (f = 1/T). Exempel:

```
10 T=TIME(12)
20 ?1/T
```

Programmet medför att periodtiden på ingång B mäts, varefter frekvensen beräknas. Även här finns en felkälla, man mäter periodtiden bara en enda gång. Det är bättre att mäta t ex 10 (eller 100) gånger, varefter datorn får beräkna medelvärdet:

```
10 FOR I=1 TO 10
20 T=T+TIME(12)
30 NEXT I
40 ?10/T
```

12.5 Mätvärdesomvandlare

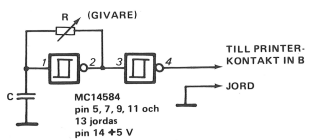
Digitala givare förekommer, t ex kodskivor för mätning av vridningsvinklar, som genererar pulståg eller andra digitala signaler. De flesta av de givare vi normalt använder är dock analoga, dvs de omvandlar mätvärdet till en analog signal. Eftersom datorer arbetar med digitala signaler, erfordras följaktligen oftast någon form av omvandling från analog till digital form, en A/D-omvandling. I det här avsnittet ska vi titta på några metoder för A/D-omvandling, som är speciellt lämpade att använda tillsammans med Esselte 100.

12.5.1 Resistans/frekvens-omvandling

Flera intressanta fysikaliska storheter går att mäta med billiga resistiva givare:

Storhet	Givare
Vridningsvinkel	Potentiometer
Läge	Skjutpotentiometer
Temperatur	Termistor
Belysning	Fotomotstånd

Det finns många andra typer av resistiva givare, t ex trådtöjningsgivare och s k Pt100-termometrar, men dessa uppvisar mycket små resistansförändringar då mätstorheten varierar och fordrar förhållandevis avancerad signalbehandling för att ge användbara signaler. Givarna i tabellen ger däremot stora ändringar i resistans för olika mätvärden och medger mycket enkel signalomvandling. En RC-oscillator är allt som behövs:



Frekvensen ges approximativt av uttrycket

$$f = \frac{3}{R\,C}$$

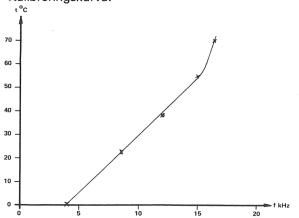
där f uttrycks i Hz, R i Ω och C i F. T ex R = 10 kΩ och C = 10 nF ger f = 30 kHz. R bör väljas större än ca 5 kΩ, annars belastas CMOS-kretsen MC14584 för mycket.

Enligt avsnitt 12.4 är det mycket lätt att mäta frekvens med Esselte 100. Visserligen blir, med denna signalomvandlare, frekvensen omvänt proportionell mot resistansen, men datorn kan om så önskas invertera mätvärdet innan det visas. Värre är i så fall att vissa billiga givare är långt ifrån linjära! Detta gäller bl a termistorer och fotomotstånd, men här kan man utnyttja datorns beräkningskapacitet för linearisering. Enklast görs lineariseringen genom att givaren kalibreras i några punkter, varefter en kurvanpassning görs.

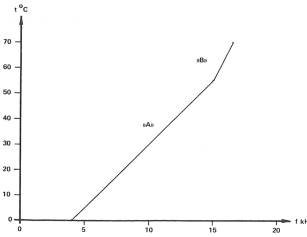
Låt oss se på ett exempel! Koppling enligt ovanstående schema, R utgjordes av en termistor. Följande kalibrering gjordes:

Temperatur °C	Frekvens kHz
1	3,99
23	8,50
38	11,97
55	14,99
70	16,40

Kalibreringskurva:



Kurvanpassningen kan göras t ex med "minstakvadratmetoden", men detta blir ganska krångligt (om man inte skriver ett program för Esselte 100 som gör beräkningarna, förstås!) Ofta blir det enklare om man betraktar kurvan som styckvis linjär. I exemplet ovan kan kurvan lämpligen representeras av två räta linjer:



Kurvan "A" har ekvationen t = 5f − 20 och "B" t = 10f − 95 (erhålls ur räta linjens ekvation). f uttrycks i kHz.

Ett program, som hämtar in data från termistorn och direkt anger temperaturen i °C, kan då se ut så här:

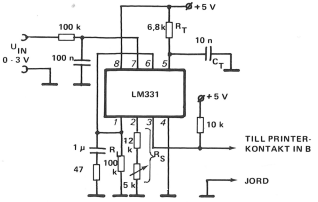
```
20 CLR
20 F=TIME(8)*0.001
30 IF F<15 THEN T=5*F-20:GOTO 50
40 T=10*F-95
50 ?T
60 GOTO 20
```

Multiplikationen med 0,001 på rad 20 görs för att få F i kHz. Ett nytt mätvärde erhålls varje tiondels sekund, vill man mäta långsammare kan en WAIT-instruktion tillfogas på rad 55. T ex WAIT(5000) får datorn att vänta 5 sekunder innan nästa mätning.

12.5.2 Spänning/frekvens-omvandling

Många givare har utsignal i form av spänningar, och även i andra sammanhang kan man ha nytta av ett gränssnitt som omvandlar spänningar till digitala signaler. Spänning/frekvens-omvandlare, eller kortare V/f-omvandlare, gör om mätvärdet till ett pulståg, vars frekvens är proportionell mot den spänning som utgör mätvärdet.

Sådana V/f-omvandlare finns som tillbehör till Esselte 100 (t ex VF100 och AD100), men om man vill kan man själv bygga ihop en V/f-omvandlare:



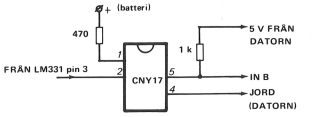
För kretsen gäller sambandet

$$f = \frac{U_{IN} \cdot R_S}{2{,}09 \cdot R_L \cdot R_T \cdot C_T}$$

f erhålls i Hz, om U_{IN} uttrycks i V, resistanserna i och kapacitansen i F. De frekvensbestämmande komponenterna (de som ingår i uttrycket för f) ska vara av god kvalitet. Använd 1% metallfilmsmotstånd och plastfoliekondensatorer! Trimpotentiometern används för fintrimning av frekvensen vid kalibrering. Enligt fabrikanten är lineariteten mycket god, typiskt 0,03%, dvs den motsvarar ungefär en s k 3 1/2 siffrig digitalvoltmeter (0 − 1999). Med de komponentvärden som anges i kopplingsschemat (och trimning av R_S) erhålls "känsligheten" 1 kHz/V. Med C_T = 1 nF blir den istället 10 kHz/V.

I vissa fall vill man galvaniskt isolera mätobjektet från datorn. V/f-omvandlaren går utmärkt att använda även i sådana tillämpningar. Dess strömförsörjning måste då skiljas från datorns och t ex batterimatning användas. Den visade V/f-omvandlaren drar bara ca 4 mA och kan matas med 4 V − 40 V. Ett litet 9 V-batteri passar alltså bra och räcker förhållandevis länge. Vidare måste utsignalen och V/f-omvandlarens jord isoleras från datorn, något som går att göra på flera sätt, bl a genom kapacitiv koppling, genom att använda en pulstransformator eller med hjälp av optokopplare. Här är normalt optokopplare det givna valet, p g a extremt god isolering (informationen överförs med ljuspulser) och billigare pris (mindre än tio kronor).

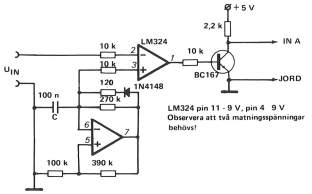
Optokopplaren ansluts så här till V/f-omvandlarens utgång:



Strömförbrukningen från batteriet ökar med någon mA. Isolationsspänningen är här 1500 V, men det finns billiga optokopplare som klarar upp till 5000 V!

12.5.3 Pulsbreddsomvandling

V/f-omvandlare fordrar en förhållandevis lång mättid. Har man krav på större snabbhet, kan en pulsbreddsomvandlare, PWM-omvandlare ("Pulse Width Modulation"), vara en bättre lösning. PWM-omvandlaren gör om mätvärdet (i vårt fall en spänning) till ett pulståg med konstant frekvens, där pulsbredden är ett mått på insignalen. En enkel PWM-omvandlare kan byggas med t ex en fyrdubbel operationsförstärkare LM324:



Med de angivna komponentvärdena blir insignalområdet ca ±100 mV och frekvensen 100 Hz. Pulsbredden kan varieras mellan 0 och 10 ms. Med Basic-instruktionen TIME(4) kan Esselte 100 mäta pulsbredden med upplösningen 100 µs, dvs 1% upplösning.

Principen är att U_{IN} jämförs med en sågtandsspänning (pin 6). Komparatorn slår om när spänningarna är lika och ger ett pulståg på utgång 1. Linearitetsfelet är några procent och beror på att sågtandsspänningen inte är helt linjär.

Den här omvandlaren passar bra när man vill plotta kurvformen hos en spänning på bildskärmen. Upplösningen är lagom stor och linearitetenn tillräcklig. Skrivs programmet i Basic, blir sveptiden ca 3 sekunder. Ett program, som plottar insignalen på bildskärmen, kan se ut så här:

```
10 CLR
20 FOR I=0 TO 95
30 PLOT(I,TIME(4)*10000)
40 NEXT I
```

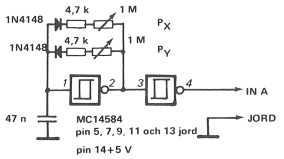
Pulstiden kan mätas tio gånger noggrannare, om man tillåter maskinkodsprogram. I så fall kan C minskas med en faktor 10, till 10 nF, varvid frekvensen ökar till 1 kHz. Varje mätning tar följaktligen 1 ms.

Givetvis kan även PWM-omvandlaren skiljs galvaniskt från datorn med hjälp av optokopplare, precis på samma sätt som visades för V/f-omvandlaren.

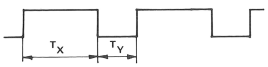
12.5.4 Joy-stick

En "joy-stick" består av två potentiometrar, som båda manövreras av en och samma spak. Den ena potentiometern känner av läget i X-led, den andra i Y-led. Joy-sticks är mycket vanliga vid s k TV-spel, men kan även användas i mera seriösa sammanhang för att mata in information i datorn.

Potentiometerlägena kan lätt omvandlas till pulsbredder, som datorn kan mäta med några Basic-instruktioner. Exempel på koppling:



Utsignalen blir ett pulståg:



Tiden T_X bestäms av potentiometern P_X, T_Y av P_Y. T_X och T_Y kan varieras mellan 75 μ och 150 ms. Ett Basic-program, som hämtar in T_X och T_Y, uttryckta som heltal X och Y mellan 1 och 150, ser ut så här:

```
10 X=INT(TIME(4)*1E4)
20 Y=INT(TIME(3)*1E4)
```

12.5.5 Exempel på mätsystem

Ett komplett datainsamlingssystem, som tar in mätvärden från 8 olika givare, kan lätt byggas upp med ett fåtal kretsar enligt schemat längst ned på denna sida.

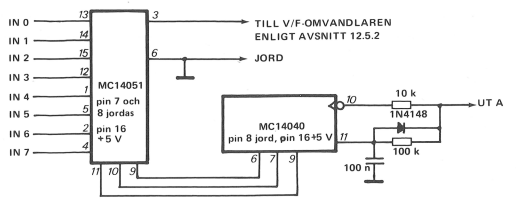
MC14051 används som multiplexer, dvs den väljer ut en av de åtta insignalerna och matar in den på V/f-omvandlaren. Multiplexern kan hantera insignaler i intervallet 0—5 V. Vilken ingång som väljs, bestäms av räknaren MC14040. Önskad ingång väljs med kommandot APULS och de båda instruktionerna OUT(1,20):OUT(N,1), som först ger en 20 ms lång puls på utgång A (för nollställning av räknaren), följd av N st 1 ms långa pulser, som stegar fram räknaren.

Vi förutsätter att V/f-omvandlarens känslighet är 10 kHz/V (enligt avsnitt 12.5.2). Följande program gör ett antal mätserier, i den ordning som DATA-satsen på rad 120 ger, dvs börjar med ingång 1, tar sedan ingång 7 osv. Det sista data, —1, anger att en mätserie är slut. Datorn väntar då 5 sekunder, gör sedan en ny mätserie osv.

```
10 CLR
20 RESTORE
30 READ N
40 APULS
50 IF N=0 THEN OUT(1,20):GOTO 80
60 IF N=-1 THEN WAIT(5000):?:
   GOTO 20
70 OUT(1,20):OUT(N,1)
80 WAIT(50)
90 F=TIME(8)*1E-4
100 ?"KANAL";N;"=";F;"VOLT"
110 GOTO 30
120 DATA 1,7,5,3,0,-1
```

Det går inte att sätta N=0 i OUT(N,1), då erhålls 256 pulser! På rad 50 särbehandlas därför fallet N=0. På rad 80 är inlagt en 50 ms fördröjning, för att V/f-omvandlaren ska hinna med att svänga in sig till det nya mätvärdet, innan mätningen börjar på rad 90. Multiplikationen med 0,0001 på rad 90 ger mätvärdet direkt i volt.

Schema för datainsamlingssystem.

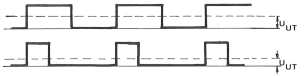


12.6 Analoga signaler

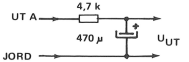
Om datorn ska användas för styrning av analoga förlopp, t ex hastigheten hos en motor, fordras en digital/analogomvandling, D/A-omvandling, av datorns utsignal. Vi ska här titta på två olika metoder, som båda använder utsignalerna på PRINTER-kontakten, den ena utnyttjar pulsbreddsmodulering, den andra en räknare och ett resistansnät.

12.6.1 Pulsbreddsmodulerad D/A-omvandlare

Datorn får generera ett pulståg med konstant frekvens, där pulsbredden är proportionell mot den önskade utsignalen. Pulståget filtreras i ett RC-nät (lågpassfilter), varvid en likspänning U_{UT} proportionell mot pulsbredden erhålls.



Kopplingsschema:



Program:

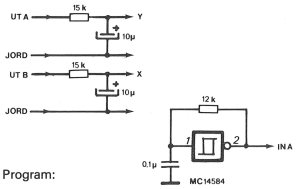
```
10 T=30
20 N=50
30 FOR I=1 TO 10*T
40 POKE(51209,60):WAIT(N+1)
50 POKE(51209,52):WAIT(101-N)
60 NEXT I
```

T är tiden (i sekunder) under vilken spänningen ska matas ut. N är ett mått på spänningen, N ska vara ett heltal mellan 0 och 100. N=100 motsvarar ungefär 3 V. Instruktionen POKE(51209,60) sätter en "etta" på utgång A. För att nollställa utgång A används POKE(51209,52).

Metoden är mycket enkel, men en nackdel är att den är långsam, det tar lång tid att ändra utspänningen p g a den stora tidskonstanten i RC-nätet (ca 25 s).

En annan nackdel med den här metoden är att datorn blir upplåst av D/A-omvandlingen, den kan inte användas för något annat ändamål så länge som den genererar den analoga utsignalen. Detta finns det dock bot för, genom att utnyttja datorns avbrottsmöjlighet kan man låta datorn ägna sig åt annan verksamhet större delen av tiden, och åt generering av pulståg under mycket kort tid. Ur användarens synvinkel ser det då ut som om datorn jobbar med två uppgifter samtidigt. Det behövs en yttre krets för att generera avbrotten, t ex en enkel oscillator. Nedanstående BASIC-program skriver egentligen in en kort maskinkodsrutin i datorns minne, en maskinkodsrutin som anropas en gång för varje avbrott, dvs en gång per period hos den yttre oscillatorn. Programmet matar ut två pulståg, ett på UT A och ett på UT B, och styr följaktligen två D/A-omvandlare samtidigt. En typisk användning är plottning av grafer på en X-Y-skrivare. Den yttre oscillatorn ansluts till IN A. Lämplig oscillatorfrekvens är ca 2 kHz. Datorn "offrar" då 15 ms per sekund åt att hålla igång de två D/A-omvandlarna, resten av tiden utnyttjas till det ordinarie BASIC-programmet.

Kopplingsschema:



Program:

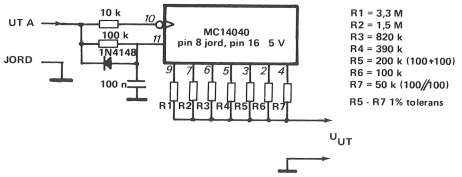
```
10 FOR I=9 TO 44
20 READ A
30 POKE(I,A)
40 NEXT I
50 DATA 95,150,5,155,6,89,151,6
60 DATA 88,88,88,203,53,247,200
70 DATA 9,95,150,7,155,8,89,151
80 DATA 8,88,88,88,203,52,247
90 DATA 200,11,246,200,8,59
100 POKE(68,128):!TILLÅT IRQ
110 POKE(51209,53):! FRÅN A IN
```

Det program som ska utnyttja D/A-omvandlarna ska placera X-koordinaten (0≥X≥255) i adressen 5 och Y-koordinaten (0≥Y≥255) i adressen 7. Detta görs med POKE-instruktioner, dvs POKE (5,X) och POKE(7,Y).

Vill man förhindra avbrott igen görs detta med kommandot POKE(51209,52).

12.6.2 Resistansnät

En snabbare D/A-omvandling erhålls med ett resistansnät och en räknare:



Omvandlaren i kopplingsschemat ger 7 bitars upplösning, dvs bättre än 1%. Utimpedansen är ganska hög, ska en lågohmig lst drivas behövs någon form av impedansomvandling, t ex en emitterföljare eller en bufferförstärkare. Inget hindrar att man gör denna omvandlartyp med högre upplösning, men då köper man lämpligen en färdig D/A-omvandlare och matar den från räknaren. På marknaden finns idag att tillgå D/A-omvandlare med upp till 16 bitar till överkomliga priser.

Program:

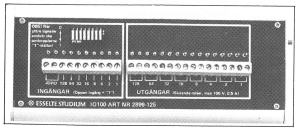
```
10 N=50
20 APULS
30 OUT(1,20):OUT(N,1)
```

N ska vara ett heltal mellan 1 och 127, U_{UT} är proportionell mot N. En omvandling tar maximalt 0,25 sekunder.

Har man krav på ännu snabbare omvandling, kan man ansluta ett resistansnät eller en kommersiellt tillgänglig D/A-omvandlare till parallellporten, som avhandlas i kapitel 11.

12.7 IO100, ett digitalt gränssnitt till Esselte 100

Som tillbehör till Esselte 100 finns ett digitalt gränssnitt, IO100, med 8 ingångar och 8 reläutgångar. IO100 kopplas till PRINTER-kontakten och är avsedd för styrning av t ex pneumatiska system. Utgångarna indikeras av lysdioder. Reläerna tål 100 V, 0,5 A, max 15 W. Ingångarna är skyddade för spänningar upp till 30 V och går att använda dels för slutande/brytande kontakter (t ex mikroswitchar), dels för normala TTL-nivåer. Parallellt med ingångarna ligger en 8-polig DIP-switch, för simulering av insignaler.



Programexemplet visar hur man hämtar in information från de 8 ingångarna och matar ut denna information på de 8 utgångarna:

```
10 IO
20 A=INP(0)
30 OUT(0,A)
```

Styrprogram och styrsekvenser blir mycket enkla, tack vare datorns speciella in/ut-instruktioner INP(0) och OUT(0,A), tidsfördröjningen WAIT(T) och de logiska funktionerna AND(X,Y), OR(X,Y) och XOR(X,Y).

Argumentet 0 i INP(0) och OUT(0,A) anger egentligen kanalnummer. Datorn är förberedd för större I/O-enheter, där kanalnummer mellan 0 och och 255 kan användas. IO100 innehåller dock ingen kanalvalslogik.

Appendix

MC 6800 INSTRUCTION SET

The MC6800 instructions are described in detail in the M6800 Programming Manual. This Section will provide a brief introduction and discuss their use in developing MC6800 control programs.

The instruction set is shown in summary form in Figure 1-3-1. Microprocessor instructions are often divided into three general classifications: (1) memory reference, so called because they operate on specific memory locations; (2) operating instructions that function without needing a memory reference; (3) I/O instructions for transferring data between the microprocessor and peripheral devices.

In many instances, the MC6800 performs the same operation on both its internal accumulators and the external memory locations. In addition, the M6800 interfaces adapters (PIA and ACIA) allow the MPU to treat peripheral devices exactly like other memory locations, hence, no I/O instructions as such are required. Because of these features, other classifications are more suitable for introducing the MC6800's instruction set: (1) Accumulator and memory operations; (2) Program control operations; (3) Condition Code Register operations.

CONDITION CODE REGISTER OPERATIONS

The Condition Code Register (CCR), also called the Program Status Byte, will be described first since it is affected by many of the other instructions as well as the specific operations shown in Figure 1-3.1-2. The CCR is a 6-bit register within the MPU that is useful in controlling program flow during system operation. The bits are defined in Figure 1-3.1-1.

b_5	b_4	b_3	b_2	b_1	b_0
H	I	N	Z	V	C

H = Half-carry; set whenever a carry from b_3 to b_4 of the result is generated by ADD, ABA, ADC; cleared if no b_3 to b_4 carry; not affected by other instructions.

I = Interrupt Mask; set by hardware or software interrupt or SEI instruction; cleared by CLI instruction. (Normally not used in arithmetic operations.) Restored to a zero as a result of an RTI instruction if I_m stored on the stacked is low.

N = Negative; set if high order bit (b_7) of result is set; cleared otherwise.

Z = Zero; set if result = 0; cleared otherwise.

V = Overlow; set if there was arithmetic overflow as a result of the operation; cleared otherwise.

C = Carry; set if there was a carry from the most significant bit (b_7) of the result; cleared otherwise.

FIGURE 1-3.1-1. Condition Code Register Bit Definition

ACCUMULATOR AND MEMORY OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	●	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	●	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	●	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A + M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B + M → B	●	●	↕	↕	R	●
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A • M	●	●	↕	↕	R	●
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B • M	●	●	↕	↕	R	●
Clear	CLR							6F	7	2	7F	6	3				00 → M	●	●	R	S	R	R
	CLRA													4F	2	1	00 → A	●	●	R	S	R	R
	CLRB													5F	2	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	●	●	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	●	●	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M → M	●	●	↕	↕	R	S
	COMA													43	2	1	A → A	●	●	↕	↕	R	S
	COMB													53	2	1	B → B	●	●	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	●	●	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	●	●	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	●	●	↕	↕	①	②
Decimal Adjust, A	DAA																Converts Binary Add. of BCD Characters into BCD Format	●	●	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	●	●	↕	↕	④	●
	DECA													4A	2	1	A − 1 → A	●	●	↕	↕	④	●
	DECB													5A	2	1	B − 1 → B	●	●	↕	↕	④	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A ⊕ M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B ⊕ M → B	●	●	↕	↕	R	●
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	●	●	↕	↕	⑤	●
	INCA													4C	2	1	A + 1 → A	●	●	↕	↕	⑤	●
	INCB													5C	2	1	B + 1 → B	●	●	↕	↕	⑤	●
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	●	●	↕	↕	R	●
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	●	●	●	●	●	●
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	●	●	●	●	●	●
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	●	●	●	●	●	●
Rotate Left	ROL							69	7	2	79	6	3				M	●	●	↕	↕	⑥	↕
	ROLA													49	2	1	A	●	●	↕	↕	⑥	↕
	ROLB													59	2	1	B	●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	●	●	↕	↕	⑥	↕
	RORA													46	2	1	A	●	●	↕	↕	⑥	↕
	RORB													56	2	1	B	●	●	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	●	●	↕	↕	⑥	↕
	ASLA													48	2	1	A	●	●	↕	↕	⑥	↕
	ASLB													58	2	1	B	●	●	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	●	●	↕	↕	⑥	↕
	ASRA													47	2	1	A	●	●	↕	↕	⑥	↕
	ASRB													57	2	1	B	●	●	↕	↕	⑥	↕
Shift Right, Logic.	LSR							64	7	2	74	6	3				M	●	●	R	↕	⑥	↕
	LSRA													44	2	1	A	●	●	R	↕	⑥	↕
	LSRB													54	2	1	B	●	●	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	●	●	↕	↕	R	●
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	●	●	↕	↕	↕	↕
Subract Acmltrs.	SBA													10	2	1	A − B → A	●	●	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	●	●	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	●	●	↕	↕	R	●
	TBA													17	2	1	B → A	●	●	↕	↕	R	●
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	●	●	↕	↕	R	R
	TSTA													4D	2	1	A − 00	●	●	↕	↕	R	R
	TSTB													5D	2	1	B − 00	●	●	↕	↕	R	R

FIGURE 1-3-1 MC6800 Instruction Set

INDEX REGISTER AND STACK POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BOOLEAN/ARITHMETIC OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				(X_H/X_L) − (M/M + 1)	●	●	↕	↕	⑧	●
Decrement Index Reg	DEX													09	4	1	X − 1 → X	●	●	●	↕	●	●
Decrement Stack Pntr	DES													34	4	1	SP − 1 → SP	●	●	●	●	●	●
Increment Index Reg	INX													08	4	1	X + 1 → X	●	●	●	↕	●	●
Increment Stack Pntr	INS													31	4	1	SP + 1 → SP	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				M → X_H, (M + 1) → X_L	●	●	⑨	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				M → SP_H, (M + 1) → SP_L	●	●	⑨	↕	R	●
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				X_H → M, X_L → (M + 1)	●	●	⑨	↕	R	●
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				SP_H → M, SP_L → (M + 1)	●	●	⑨	↕	R	●
Indx Reg → Stack Pntr	TXS													35	4	1	X − 1 → SP	●	●	●	●	●	●
Stack Pntr → Indx Reg	TSX													30	4	1	SP + 1 → X	●	●	●	●	●	●

JUMP AND BRANCH OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BRANCH TEST	5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	4	2										C = 0	●	●	●	●	●	●
Branch If Carry Set	BCS	25	4	2										C = 1	●	●	●	●	●	●
Branch If = Zero	BEQ	27	4	2										Z = 1	●	●	●	●	●	●
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	●	●	●	●	●	●
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	●	●	●	●	●	●
Branch If Higher	BHI	22	4	2										C + Z = 0	●	●	●	●	●	●
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	●	●	●	●	●	●
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	●	●	●	●	●	●
Branch If Minus	BMI	2B	4	2										N = 1	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	4	2										V = 0	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	4	2										V = 1	●	●	●	●	●	●
Branch If Plus	BPL	2A	4	2										N = 0	●	●	●	●	●	●
Branch To Subroutine	BSR	8D	8	2											●	●	●	●	●	●
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	●	●	●	●	●	●
Jump To Subroutine	JSR				AD	8	2	BD	9	3					●	●	●	●	●	●
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	●	●	●	●	●	●
Return From Interrupt	RTI										3B	10	1	——— ⑩ ———						
Return From Subroutine	RTS										39	5	1	See special Operations	●	●	●	●	●	●
Software Interrupt	SWI										3F	12	1		●	S	●	●	●	●
Wait for Interrupt	WAI										3E	9	1		●	⑪	●	●	●	●

CONDITIONS CODE REGISTER OPERATIONS	MNEMONIC	INHER OP	INHER ~	INHER #	BOOLEAN OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0 → C	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	●	R	●	●	●	●
Clear Overflow	CLV	0A	2	1	0 → V	●	●	●	●	R	●
Set Carry	SEC	0D	2	1	1 → C	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	●	S	●	●	●	●
Set Overflow	SEV	0B	2	1	1 → V	●	●	●	●	S	●
Acmltr A → CCR	TAP	06	2	1	A → CCR	——— ⑫ ———					
CCR → Acmltr A	TPA	07	2	1	CCR → A	●	●	●	●	●	●

LEGEND:

OP	Operation Code (Hexadecimal);	00	Byte = Zero;
~	Number of MPU Cycles;	H	Half-carry from bit 3;
#	Number of Program Bytes;	I	Interrupt mask
+	Arithmetic Plus;	N	Negative (sign bit)
−	Arithmetic Minus;	Z	Zero (byte)
•	Boolean AND;	V	Overflow, 2's complement
M_{SP}	Contents of memory location pointed to be Stack Pointer;	C	Carry from bit 7
		R	Reset Always
		S	Set Always
+	Boolean Inclusive OR;	↕	Test and set if true, cleared otherwise
⊕	Boolean Exclusive OR;	●	Not Affected
M	Complement of M;	CCR	Condition Code Register
→	Transfer Into;	LS	Least Significant
0	Bit = Zero;	MS	Most Significant

CONDITION CODE REGISTER NOTES:
(Bit set if test is true and cleared otherwise)

① (Bit V) Test: Result = 10000000?
② (Bit C) Test: Result = 00000000?
③ (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
④ (Bit V) Test: Operand = 10000000 prior to execution?
⑤ (Bit V) Test: Operand = 01111111 prior to execution?
⑥ (Bit V) Test: Set equal to result of N ⊕ C after shift has occurred.
⑦ (Bit N) Test: Sign bit of most significant (MS) byte of result = 1?
⑧ (Bit V) Test: 2's complement overflow from subtraction of LS bytes?
⑨ (Bit N) Test: Result less than zero? (Bit 15 = 1)
⑩ (All) Load Condition Code Register from Stack. (See Special Operations)
⑪ (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
⑫ (ALL) Set according to the contents of Accumulator A.

FIGURE 1-3-1 (continued)

REGISTER

alfanumeriska tecken 31
användardefinierade tecken 32
APULS 38, 47
avbrott 12
AXIS 31
bandspelare 27
BAR 29

Basic-program, lagring av 11
Basic-tolk 9
 minnesplan 9
 tider 12
 subrutiner 40—41
BAS100 44—46
beräkningsrutiner 41
bildskärm 17
BPULS 38, 47

CLR 18
CPU 1
CTRL 17
CTRL-H 9
CTRL-M 9
CTRL-π 21
CTRL-X 9
CUR 18

databaser 43
D/A-omvandlare 57

format (bandspelare) 27
format (utsignal) 21
formatändring 22
frekvensmätning 53

GCLR 18, 47
gemener 19
grafik 29—35
grafikhantering 29
grafikkommandon 29
grafik, rörlig 34
GRAFISKA RUTINER I Basic-tolken 41
grafiska tecken 30
GRCL 18
gränssnitt, digitalt 58
gränssnitt för fototransistor 52
gränssnitt för frekvensmätning 53
GSTO 18

hastighet för Basic-tolken 12
hsqtighet för överföring 16
hastighet för printer 21

INP 38
insignaler 47
Interface-kontakter 21—22
Interrupt 12
IO 38
IO100 58
I/O-del 3

Joy-stick 56

komponentförteckning 3
komponentplacering 6—7
kontrollkoder 17
kurvor, plottning av 29
kurvprickar 31
kruvritning 31

linjedragning 19

markör 17
maskinkodsmonitorn 38
maskinkodsprogram 37, 40
maskinspråk 37
MASTER 15, 16
MASTER-mod 25
MC6802 1
mikroprocessor 1, 37
minne, hushållning med 13
minnen 2
minneskarta 37
minnesplan för Basic-tolk 9
modulator 2
monitorkommandon 39
MOVE 18
mätning 51
mätsystem 56
mätvärdesinsamling 44
mätvärdesomvandlare 53

NET100 25

OUT 38, 47

parallellgränssnitt 51
PIA 37
PLOT 29
PORT 15
portsystemet 15
POS 18
PRINT # 18
printer 21
PRINTER-kontakten 47, 51
PROM 15
PROM-kassetter 15
pulsbreddsomvandling 55
punktfigurer 31

radbuffert 9
realtidsklocka 48
register 43
RESET 11
resistansnät 58

seriegränssnitt 51
SIZE 18, 32
skrivare 21
små bokstäver 19
Software Interrupt 12
spänning—frekvens 54
staplar 29
statusceller 13
stora tecken 32
strömförsörjning 2
Studium FC 52
styrning 51
styrning av bandspelare 27
symboler 30, 31
SYS 37
systemswitchar 13

TAB 18
tabellritning 19
tangentbord 4, 18
teckentabell 30
teckenuppbyggnad 31
TERM 21, 22

terminalfunktion 22
 till annan Esselte 100 25
Texas Silent (printer) 21
texthantering 19
tidmätning 51
TIME 38, 51, 52
Trendcom/Esselte (printer) 21

upplösning vid grafik 29
USER 37
utsignaler 47

variabler, lagring av 11
versaler 19
video 3
Video-RAM 17
V/f-omvandlare 54

yttre terminal 16

överföring av program 26
överföringshastighet 16

TEMAL 100

Teknisk manual ESSELTE 100

TEMAL 100 är en teknisk beskrivning av mikrodatorn ESSELTE 100.

Boken beskriver hårdvarans uppbyggnad och konstruktion, BASIC-tolken, hur man kommunicerar med ESSELTE 100, hur man kopplar ihop flera ESSELTE 100, hur man lägger upp databaser, hur grafiken fungerar, hur ESSELTE 100 används för mätning och styrning samt mycket annat. Ett stort avsnitt ägnas åt programmering i maskinspråk, och adresser anges till subrutinerna i BASIC-tolken.

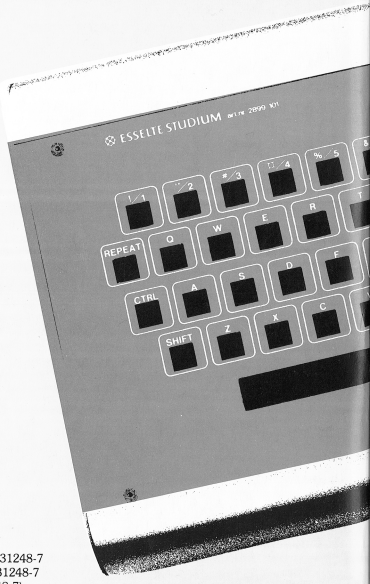
Andra böcker om ESSELTE 100:

ESSELTE 100 MANUAL
(medföljer maskinen, kan också beställas separat)

BASIC till 100
(lärobok i BASIC)

BASIC till 100 ÖVNINGSBOK
(övningsuppgifter till läroboken)

LÄRARHANDLEDNING BASIC till 100



ISBN 91-24-31248-7
Best nr 24-31248-7
(24-31248-7)

ESSELTE STUDIUM

Norra regionen O. Rådhusgatan 6, 902 46 UMEÅ ☎ 090-13 70 00 *Östra regionen* 171 76 SOLNA ☎ 08-734 30 00
Västra regionen Box 3075, 400 10 GÖTEBORG ☎ 031-17 99 30 *Södra regionen* Box 4086, 203 11 MALMÖ ☎ 040-770 70